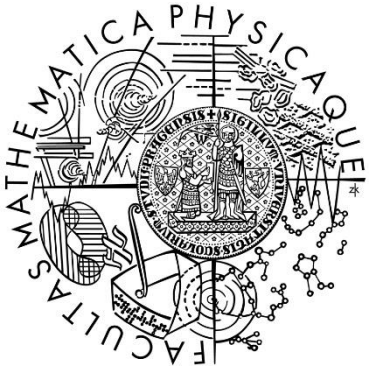
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



René Fischer

Správa a vizualizace ontologií

Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Michal Kopecký Ph.D.

Studijní program: Informatika

Studijní obor: softwarové systémy

Praha 2014

[Vzor: Vevázaný list – kopie podepsaného „Zadání diplomové práce“. **Toto zadání NENÍ součástí elektronické verze práce NESKENOVAT**.]

Touto cestou by som rád poďakoval vedúcemu práce pánovi RNDr. Michalovi Kopeckému Ph.D. za odborné vedenie, cenné rady, priebežnú kontrolu výsledkov a trpezlivosť počas písania práce. Ďalej by som rád poďakoval všetkým, ktorí ma podporovali pri písaní tejto práce, obzvlášť rodičom, blízkym priateľom ale aj zamestnávateľovi za časovú ústretovosť.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 10. 04. 2014 …….

Název práce: Správa a vizualizace ontologií

Autor: René Fischer

Katedra / Ústav: Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Michal Kopecký Ph.D., Katedra softwarového inženýrství

Abstrakt: Moderní informační technologie se neustále vyvíjejí a jejich vývoj je neoddělitelně spjat s vývojem internetu. Množství dat a informací publikovaných a dostupných na internetu každoročně roste v jednotkách Exa-bajtů. Vývoj sémantického webu má za cíl pomocí přidání tzv. sémantických značek napomoci strojům, a v konečném důsledku i člověku, vyznat se a profitovat z informací nacházejících se v těchto (nejen) na internetu uložených datech. Sémantická značka dává prvku jí označenému strojově uchopitelný význam. Tento význam spočívá v pozici a vztazích dané značky k jiným značkám v rámci nějaké množiny (slovníku) takovýchto značek. Ontologií nazýváme množinu nebo slovník takových značek, která zachycuje specifickou znalost z určité oblasti života, technologie nebo problematiky. Práce si klade za cíl v teoretické části prozkoumat historii vývoje reprezentace znalostí, vývoje ontologií a ontologického inženýrství a technologie sémantického webu. V praktické části analyzuje jednu z existujících velkých a reálné používaných ontologií SNOMED CT a navrhuje nástroj, který by mohl přispět k rozšíření a zjednodušení práce na vývoji ontologií. Posledním záměrem práce je implementovat zvolenou část navrhnuté aplikace.

Klíčová slova: ontologie, správa, vizualizace

Title: Ontology Management and Visualization

Author: René Fischer

Department: Department of Software Engineering

Supervisor: RNDr. Michal Kopecký Ph.D., Department of Software Engineering

Abstract: Modern information technologies are unstoppably advancing and their development is inseparably linked with the development of the Internet. The amount of data and information published and available on the Internet is growing in units of Exabyte every year. Using so-called semantic tags added to documents, the Semantic Web movement aims to help machines, and eventually man, to benefit from the information contained in those (not only) on the Internet stored data. Elements marked with a semantic tag is given a machine- recognizable information. The importance of it lies in the positions and relationships of the tag to other tags within some set of (dictionary) of such tags. Ontologies are sets or dictionaries of such marks that capture specific knowledge of certain areas of life, technology or problems. The theoretical part of this work aims to explore the history and the development of knowledge representation, the development of ontologies and ontological engineering and semantic web technologies. In the practical part we analyze the SNOMED CT ontology, what is a large actually used ontology. We also design a tool that should help to enhance and simplify the development of ontologies. In the last, a plan is to implement a part of the designed tool.

Keywords: ontology, management, visualization

Názov práce: Správa a vizualizácia ontológií

Autor: René Fischer

Katedra / Ústav: Katedra softwarového inženýrství

Vedúci diplomovej práce: RNDr. Michal Kopecký Ph.D., Katedra softwarového inženýrství

Abstrakt: Moderné informačné technológie nezastaviteľne napredujú a ich vývoj je neoddeliteľne spätý s vývojom internetu. Množstvo dát a informácií publikovaných a dostupných na internete každoročne rastie v jednotkách Exa-bajtov. Hnutie sémantického webu chce pomocou pridávania tzv. sémantických značiek pomôcť strojom, a v konečnom dôsledku človeku, vyznať sa a profitovať z informácií nachádzajúcich sa v týchto (nie len) na internete uložených dátach. Sémantická značka dáva prvku ňou označeným strojovo uchopiteľný význam. Tento význam spočíva v pozícií a vzťahmi danej značky k iným značkám v rámci nejakej množiny (slovníka) takýchto značiek. Ontológiou nazývame množinu alebo slovník takých značiek, ktorá zachytáva špecifickú znalosť z určitej oblasti života, technológie alebo problematiky. Práca si kladie za cieľ v teoretickej časti preskúmať históriu vývoja reprezentácie znalostí, vývoja ontológií a ontologického inžinierstva a technológie sémantického webu. V praktickej časti analyzujeme jednu z existujúcich veľkých a reálne používaných ontológií SNOMED CT a navrhujeme nástroj, ktorý by mal prispieť k rozšíreniu a zjednodušeniu práce na vývoji ontológií. Posledný zámer práce je implementovať nejaký časť navrhnutej aplikácie.

Kľúčové slová: ontológia, správa, vizualizácia

Obsah

1. Úvod

Ako študijný odbor som si na MFF vybral databázové systémy. Žil som v naivnej predstave, že ak viem *SQL*[1], ak ovládam *E-R modelovanie*[2], ak viem ako to funguje vo vnútri spracovania *SELECT*[3] dotazu a ukladanie dát v *relačných databázach*[4], som „*databázový špecialista*“, ktorého nemôže nič prekvapiť a že sa s touto, sem tam aktualizovanou znalosťou budem môcť za spomínaného špecialistu považovať stále.

K prvému uvedomovaniu mojej naivity mi pomohlo jedno náhodou zahliadnuté video s názvom *„Viete o tom“*[5], kde autori videa tvrdia (možno kúsok prehnane), že : *“Množstvo nových technických informácií sa zdvojuje každé 2 roky, čo pre študentov začínajúcich 4 ročné štúdium technológií znamená, že polovica toho čo sa budú učiť v prvom roku štúdia bude zastarané počas ich tretieho roku štúdia.”*[6].

Ako druhý popud mi poslúžila prednáška z predmetu *Databázové systémy v praxi* s pánom RNDr. Ondrejom Zýkom, ktorý nám na tabuľu zakreslil časovú líniu databázových technológií, s ktorými sa počas svojich štúdií a dlhoročnej kariéry databázového špecialistu stretol. Žiaľ, z hlavy ju už reprodukovať neviem, ale podstatný fakt, ktorý som si zapamätal bol ten, že databázové „*main-streamové*“ technológie sa pravidelne v cca 10-15 ročných cykloch menili, pričom cyklus relačných databáz bol podľa tohto predpokladu práve končiaci cyklus[7]. Čo by malo nasledovať potom sme už nerozoberali, resp. pán Zýka odvetil, že o tom ten predmet nie je.

---

[1] Structured Query Language: http://en.wikipedia.org/wiki/SQL

[2] Entity-relationship modeling: http://en.wikipedia.org/wiki/Entity-relationship_model

[3] SQL SELECT statement: http://www.w3schools.com/sql/sql_select.asp

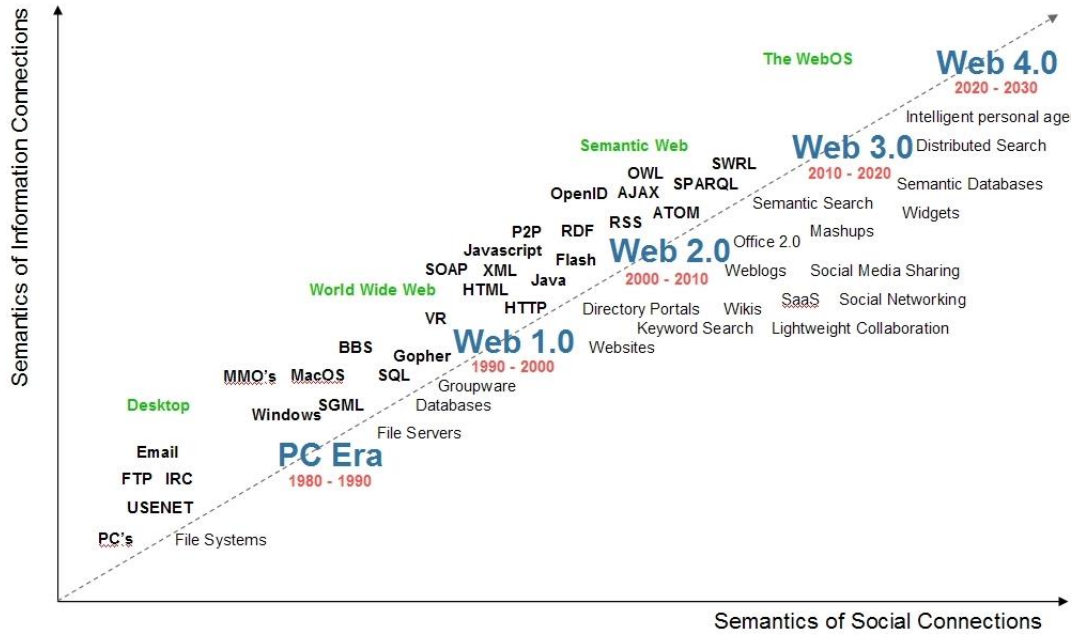
[4] Relačné databáze: http://en.wikipedia.org/wiki/Relational_database

[5] Originálny názov: *„Did You Know“*. Dostupné napr. http://www.youtube.com/watch?v=3vApOQb8A8k

[6] Anglický originál: *“The amount of new technical information is doubling every 2 years... For students starting a 4 year technical degree this means that... half of what they learn in their first year of study will be outdated by their third year of study”*

[7] Netvrdím tým, že by vývoj relačných databáz bol úplne u konca. Popredný hráči na trhu ako Oracle, SAP a Microsoft stále investujú do ich vývoja a rozvoja ako napr. v smere rýchlosti spracovania (*in-memory*) alebo špecializácii na výpočty agregácií (*column-oriented*)

Obrázok 1: Vývoj informačných (webových) technológií
Zdroj: http://novaspivack.typepad.com/RadarNetworksTowardsAWebOS.jpg



Vývoj v databázových technológiách je neoddeliteľne spätý s vývojom internetu (ďalej ako web), resp. webových technológií. Nejedna štúdia (víz. Obrázok 1) vývoja informačných technológií, s dôrazom na webové technológie, označuje pre aktuálne a následné desaťročie ako „*main-streamové*“ technológie okolo sémantického webu, ktorého hlavným cieľom je umožniť automatizované spracovanie informácií na webe strojmi. Sémantický web považujem za veľmi zaujímavú technológiu s veľkou perspektívou do budúcnosti, preto som sa rozhodol vo svojej diplomovej práci venovať práve jemu, konkrétne jednej jeho neoddeliteľnej prerekvizite, ontológiám.

1.1.Bleskovo o ontológiách

Slova prirodzeného jazyka ani jednotlivé slovné spojenia nemusia mať v danom jazyku jediný sémantický význam. Vo väčšine prípadov – ale nie vždy – dokáže ľudský mozog k danému pojmu automaticky priradiť z rady významov ten správny. Pre počítač predstavuje tato klasifikácia oveľa tvrdší oriešok. Keď napríklad užívateľ zadá do vyhľadávača slovo „*jaguár*“, tak hľadá informácie buď o vozidlách značky *jaguár*[8], alebo o mačkovitej šelme žijúcej v strednej a južnej Amerike[9]. Slovo *jaguár* pritom nie je jediné s viacerými významami. Ako teda vyhľadávač, stroj, môže vedieť čo užívateľ chce, keď pre neho je slovo „*jaguár*“ iba niekoľko za sebou idúcich písmen?

[8] Jaguar Cars: http://en.wikipedia.org/wiki/Jaguar_Cars

[9] Panthera onca: http://en.wikipedia.org/wiki/Jaguar

Nedorozumenie môže pritom mať i omnoho fatálnejšie dôsledky. Každý rok dochádza k úmrtiam a úrazom spôsobeným zlou komunikáciou medzi zdravotníkmi, nedôkladnými a nejednoznačnými lekárskymi správami, zábudlivosťou zaneprázdnených alebo unavených pracovníkov. Dodanie štandardu pre klinickú terminológiu pre použitie naprieč svetom zdravotných informačných systémov by mohlo zbytočným udalostiam predísť.

Ontológie sú (pre jednoduchosť) slovníky, ktoré slovu ako „*jaguár*“ priradí význam pochopiteľný aj pre stroje, a ktoré pomôžu výrazom „*alergia na penicilín*“ v slovenskom a „*Аллергия на пенициллин*“ v ruskom zdravotnom systéme dať rovnaký význam.

1.2.Cieľ práce

Ústrednou témou tejto práce je pojem ontológia a okolo neho sa bude točiť celá práca. Rád by som priblížil históriu a vývoj tohto pojmu, počnúc oborom zaoberajúcim sa problematikou zberu a reprezentácie znalostí. Zámer bude priblížiť aj ontologické inžinierstvo, čo je „menej známy bratranec“ toho softwarového inžinierstva. Ciele týchto oborov sú veľmi podobné a líšia sa v tom, že na miesto softwaru má výstupom procesu byť ontológia.

V praktickej časti chcem priblížiť nejakú existujúcu a reálne používanú ontológiu, preto si predstavíme jednu rozsiahlu medicínsku ontológiu, SNOMED CT. Venovať sa budeme už nie tak veľmi novej, zato na význame narastajúcej technológií, ktorá ontológie používa, sémantický web. Ten považujem za veľmi perspektívnu technológiu s teoreticky obrovským budúcim prínosom, ktorú ale stále trápia rôzne „*detské choroby*“. Medzi takéto problémy radím aj nedostatok *dobrého* programového vybavenia , ktoré by podporovalo rozšírenie tejto technológie v radoch (nie len) webových vývojárov. Myslím tým nástroje pre editáciu ontológií až po software na vývoj sémantizovaných webových stránok a dokumentov, alebo pre udržiavanie alebo anotáciu už existujúcich stránok a dokumentov. Skupinu vývojárov spomínam zámerne, pretože laická verejnosť by sa správne o nijakých zmenách na pozadí súčasného webu nemala vôbec dozvedieť, pocítiť by mala iba prínos tohto celého snaženia. Aktívna časť verejnosti (čitateľov, blogerov, autorov článkov, prispievateľov do diskusií/”*wikín*”) by sa mohla zapojiť napríklad doplňovaním sémantiky do existujúcich stránok pomocou vhodných nástrojov, z čoho by prosperovali ako ostatný čitatelia, vyhľadávače tak aj samotný vývojári.

Posledným zámerom práce je nepriamo podporiť väčšie rozšírenie tejto technológie a ontológií návrhom systému správy a vizualizácie ontológií ako je SNOMED CT a čiastočne implementovať prototyp tejto aplikácie.

1.3.Obsah kapitol

Nasledujúca kapitola 2 Sémantický web, v skratke a hrubo predstaví riešenie sémantického webu. Ide iba o jeho priblíženie, vysvetlenie jeho konceptu. Keďže práca často pracuje s pojmom ontológia a aj jej praktická časť sa venuje software pre ich vývoj, je tomuto pojmu a jeho histórií, metódam vývoja a iným venovaná celá kapitola 3 Ontológie . Do väčších technických detailov sémantického webu sa znova vrátime v kapitole 4 Štruktúra riešenia sémantického webu, kde si popíšeme jednotlivé vrstvy sémantického riešenia, tzv. *zásobník sémantického riešenia*[10] (ďalej iba ako SWS). V kapitole 5 SNOMED CT sa už zaoberáme kompletne ontológiou SNOMED CT. Venuje sa ako jej obsahu, tak jej štruktúre a vnútornej reprezentácií.

Kapitolou 6 Názor : Sémantický web a ontológie, v ktorej opisujem svoj názor na sémantický web, jeho súčasnosť, budúcnosť, jeho nedostatky, *„detské choroby“*, atď. začne praktická časť tohto diela. Vlastnosti, funkčnosti a návrh programu pre správu a vizualizáciu ontológií sú popísané v kapitole 7 Manažment systém pre ontológie. Popisuje všetko, ako funkčnosti ktoré boli, ale aj neboli implementované. Obsah kapitoly **Error! Reference source not found. Error! Reference source not found.** dostatočne popisuje jej názov.

Posledná kapitola, 8 Záver, zhŕňa celú túto prácu. Porovnáva ciele stanovené na začiatku tejto práce s cieľmi, ktoré boli skutočne dosiahnuté.

[10] Semantic Web Stack: http://en.wikipedia.org/wiki/Semantic_Web_Stack

2. Sémantický web

Súčasné potreby človeka a firiem zahŕňajú úlohy, ktoré nutne kombinujú dáta, ktoré sú uložené naprieč celým webom alebo firmou, v (ne)štruktúrovanom, proprietárnom/voľne použiteľnom a (ne)štandardnom formáte. Medzi takéto úlohy môže patriť napr.

1. integrácia dát v rámci podniku
2. dohadovanie stretnutí - transport, ubytovanie, synchronizácia kalendárov
3. získavanie informácií zo záznamov pacienta
4. získavanie informácií z neštruktúrovaných dát z rôznym účel atď.

Človek týmto informáciám rozumie a vie ako ich kombinovať aby dosiahol požadovaný výsledok. Často žiaľ narazí na bariéry ako sú cudzie jazyky, rôzne formáty alebo použité slovníky. Stroje, teda predovšetkým počítače (ďalej ako PC), takto „múdre“ nie sú a webovým stránkam (ďalej iba ako stránky)[11], ale aj iným formátom dokumentov a informačným zdrojom, rozumejú iba minimálne (víz. Tabuľka 1) a nevedia informácie kombinovať tak ako ľudia. Odpovede na naše otázky sa pritom často nenachádzajú iba na jednom mieste ale sú rozprestreté vo viacerých zdrojoch, pričom súčasný spôsob vracania výsledkov hľadania informácií poskytuje zoznam položiek z jednotlivých miest s iba čiastočnou informáciou.

Aby nám PC boli viac nápomocné v spomínaných úlohách, je potreba, aby PC viac rozumeli tomu o čom stránka informuje. Nie až tak ako tomu rozumie človek, ale tak aby ich PC vedeli spracovávať samé. Teda aby vedeli napr. :

- podstatu objektu s názvom „*Mat.-Fyz. f.*“
- čo je tá „*vec*” (*mff.cuni.cz*) na druhej strane odkazu
- vzťah medzi nimi dvoma - prečo sú spojené

Dosiahnuť tento cieľ by bolo možné 2 spôsobmi :

1. Naučiť PC rozumieť dátam - prirodzenému jazyku, obrázkom, …
   - Predstavuje multi-milionový business tvorený rôznymi typmi aplikácií, od OCR[12] po produkty spracovávajúce neštruktúrovaný text[13]

[11] Netýka sa to iba webových stránok, ale aj iných typov dokumentov a informačných zdrojov, ale pre zachovanie prehľadnosti budeme pracovať v texte iba s webovými stránkami

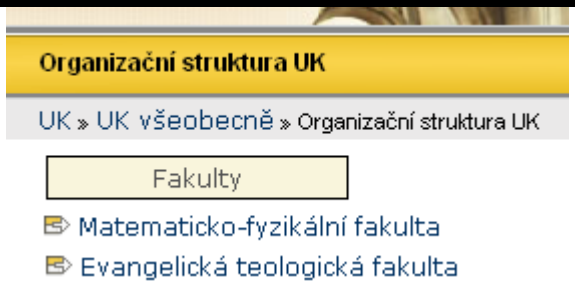
[12] Software pre prevod textu (písaný, tlačený) z obrázkov do digitálnej podoby (Optical character recognition): http://en.wikipedia.org/wiki/Optical_character_recognition

- Okrem iných sa týmto veľmi ťažkým problémom zaoberá aj umelá inteligencia

2. Spraviť dáta ľahšie porozumiteľné a spracovateľné pre PC, dať im význam
   - tomuto prístupu sa hovorí *„sémantizácia“*[14] *(webu)* .

Sémantický web je rozšírenie súčasného webu, kde informácie majú presne stanovený význam, umožňujúce PC a ľuďom lepšiu spoluprácu[15]. Jeho hlavným účelom je posunúť v pred strojovú interpretabilitu informácií nachádzajúcich sa na webe, čo vedie k zlepšeniu dostupnosti a dorozumeniu informácií ľuďom, k výmene znalostí medzi automatizovanými webovými službami a hlavne umožňuje otvorené a bezobslužné[16] zdieľanie znalosti.

Tabuľka 1: Vnímanie stránky počítačom verzus človekom
Zdroj: http://www.cuni.cz/UK-21.html

	Vidí	Rozumie
Človek		Univerzita Karlova má fakulty • Mat.-Fyz. fakulta • Evang. teol. fakulta • ... Ak chcem navštíviť stránky konkrétnej fakulty, kliknem na odpovedajúci odkaz...
PC[17]	Fakulty ... <su> Mat.-Fyz. f. ... <su>	???

2.1.K čomu to je dobré?

Google je pekný príklad, ako vyťažiť z minima sémantiky dostupnej na súčasnom webe maximum. Ide o samotný odkaz, ktorý je počítačovo spracovateľný :

---

[13] Napríklad SAS Content Categorization: http://www.sas.com/text-analytics/enterprise-content-categorization

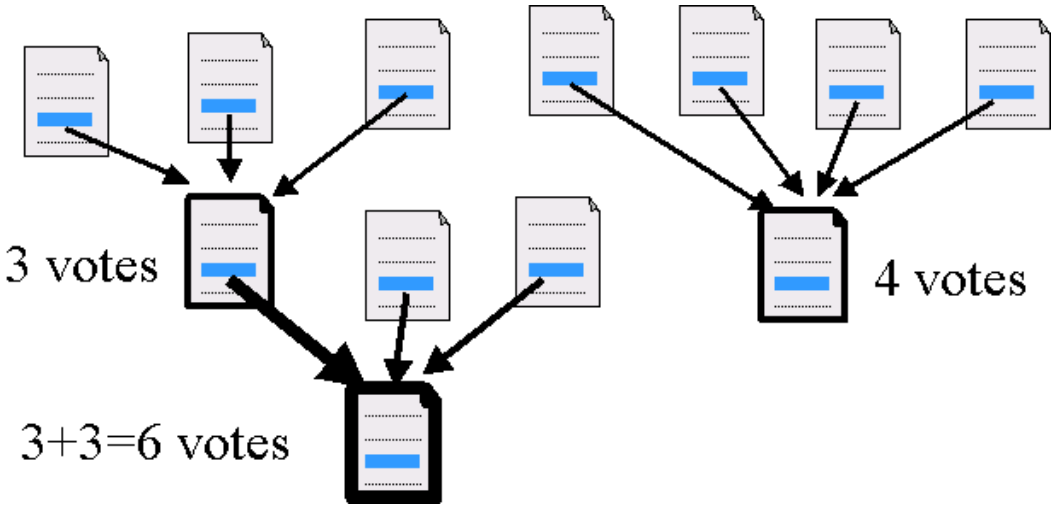
[14] Sémantika: http://cs.wikipedia.org/wiki/Sémantika

[15] Sémantický web: http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(18)

[16] unattended – nestrážené, bez dozoru, bez obsluhy, nespravované

[17] Zdrojový kód stránky v jazyku HTML: http://en.wikipedia.org/wiki/HTML

Obrázok 2: Odkazy medzi webovými stránkam
Zdroj: http://www.w3.org/2004/Talks/0612-sb-wsswapps/how-google-works.GIF



"Toto" odkazuje na "tamto" = popularita.

Tento jednoduchý princíp je základom výpočtu tzv. *PageRank*[18] stránok, ktorý vyjadruje niečo ako dôveryhodnosť alebo dôležitosť stránky[19]. Základný princíp výpočtu spočíva v tom, že čím viac stránok (a s čím vyšším *PageRankom*) odkazuje na danú stránku, tým vyšší *PageRank* je stránke pridelený (víz. Obrázok 2). Predstavte si možnosti využitia väčšej sémantiky na stránkach.

2.2.Využitie

Jedna z najfrekventovanejších činnosti človeka na súčasnom webe je vyhľadávanie informácií. Súčasne používané jazyky pre tvorbu webových stránok, HTML a CSS, sú prispôsobené pre zverejňovanie informácií pre ľudí a teda aj samotné stránky sú skôr prezentačne orientované. Stroje pri tejto štruktúre stránok iba ťažko zisťujú „*o čom*“ daná stránka je. Súčasný gigant vo vyhľadávaní na webe, *Google*, nám vie často odpovedať na naše dotazy, ale iba do obmedzenej miery.

- so zvyšujúcim sa počtom podmienok, resp. výrazov, ktoré človek zadáva do vyhľadávača sa znižuje relevantnosť odpovedí
- pýtajúci často nie je schopný (alebo ochotný) prechádzať tisíce či milióny vrátených odpovedí a zisťovať ich relevantnosť

[18] PageRank: http://en.wikipedia.org/wiki/PageRank
[19] Existujú aj iné algoritmy pracujúce nad odkazmi, napr. HITS: http://en.wikipedia.org/wiki/HITS_algorithm ale PageRank je asi najznámejší

Aj napriek tomu, že *Google*, ale aj iné spoločnosti ako je *SAS* a *IBM*, na tento účel investuje ohromné prostriedky a využíva netriviálne technológie (napr. *„text-minig"*[20], *spracovanie prirodzeného jazyka*[21]), na nasledujúce dotazy by nám odpovede hľadal ťažko:

- *Nájdi mi zdravotné stredisko s ortopedickým oddelením v blízkosti miesta mojej aktuálnej dovolenky, ktoré akceptuje cestovné poistenie od spoločnosti XY.*
    - Dotaz by ste zadali samozrejme v rodnej reči resp. vyhľadávanie by bolo nezávislé na jazyku stránok
    - Ošetrujúci lekár by nám chcel pichnúť injekciu s vakcínou XY, pričom potrebuje vedieť či nie ste na niečo alergický. Stačilo by zadať dotaz do zdravotného systému, ktorý by si poradil s jazykom doktora, tak s jazykom v ktorom sú vedené Vaše zdravotné záznamy
- *Nájdi mi obchod ktorý predáva a má na sklade výrobok X a Y, alebo 2 obchody, kde v jednom majú na sklade X a v druhom Y, ktoré by boli od seba max 500m. a nachádzali sa v Prahe 1*

Prečo je teda v súčasnej dobe poznamenanej boomom *sociálnych sietí* [22] a *„lajkovaním"*[23] nemožné položiť dotaz :

- *nájdi v mojom meste, na dnes alebo zajtra, koncert kapely, ktorú majú radi (dali jej „lajk") moji známi zo skupiny blízky priatelia*[24]

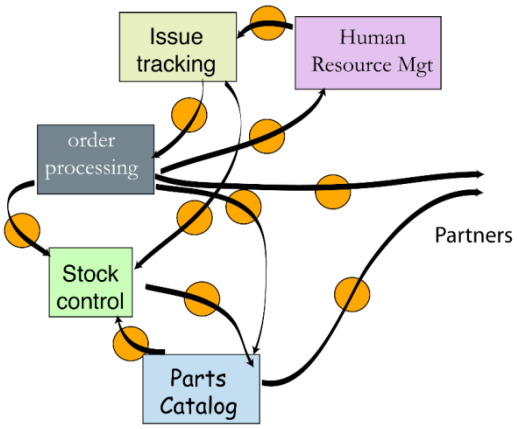
V korporáciách často funguje súčasne viac systémov, kde každý môže byť od iného dodávateľa, implementovať iný proprietárny formát, používať iný databázový systém, iný dátový model, atď. a ich prepojenie môže byť zložité, alebo finančne náročné. V súčasnosti používané riešenia

- nesystematické pridávanie spojení medzi systémami (víz. Obrázok 3)
- systematickejšie riešenie pomocou tzv. zberníc – *BUS*[25] napr. *Enterprise service bus*[26] sú často drahé proprietárne technológie (víz. Obrázok 4)

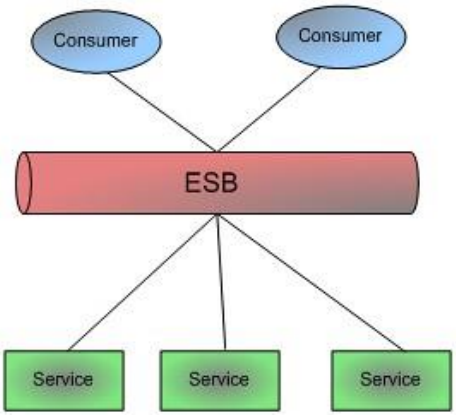
---

[20] Text-mining: http://en.wikipedia.org/wiki/Text_mining
[21] Natural language processing: http://en.wikipedia.org/wiki/Natural_language_processing
[22] Social Network: http://en.wikipedia.org/wiki/Social_network
[23] Like button: http://en.wikipedia.org/wiki/Like_button
[24] Nech *„blízky priatelia"* je špecificky označená skupina spojení na iné osoby napr. na sosiálnej sieti Facebook: http://en.wikipedia.org/wiki/Facebook
[25] BUS: http://en.wikipedia.org/wiki/Bus_(computing)

Obrázok 3: Súčasná forma firemných dát
Zdroj: http://www.w3.org/DesignIssues/diagrams/nbyn.png



Obrázok 4: Enterprise Service Bus
Zdroj: http://www.infoq.com/resource/articles/ESB-alternative/en/resources/image1.jpg



Dotazy typu *„Zisti najbližší možný termín porady s vedúcimi oddelení dodávajúcich súčiastky potrebné na montovanie výrobku XY"* by boli aj napriek použitiu ESB iba ťažko implementovateľné.

Väčšina obsahu stránok je pritom generovaná zo štruktúrovaného zdroja, je preto kúsok podivné prečo prehliadače a *crawler*[27] musia význam stránok získavať týmto zložitým a zdĺhavým spôsobom.

2.3.Ako by to bolo možné

Väčšina súčasného webu je z pohľadu PC iba orientovaný graf, kde stránky predstavujú vrcholy grafu a odkazy medzi nimi sú hrany (víz. Obrázok 6). Pričom samotné dáta vo vrcholoch grafu (na stránkach) nedávajú PC žiaden, resp. minimálny význam. Pridaním sémantických informácií do zdrojových kódov stránok by rapídne zmenilo túto situáciu a weby by sa stali oveľa viac strojovo spracovateľné (víz. Obrázok 5). Pripomenieme príklad z úvodu : *„Nájdi mi zdravotné stredisko s ortopedickým oddeleným v blízkosti miesta mojej aktuálnej dovolenky, ktoré akceptuje cestovné poistenie od spoločnosti XY"*

Pridaná sémantika by „*PC napovedala*", že :

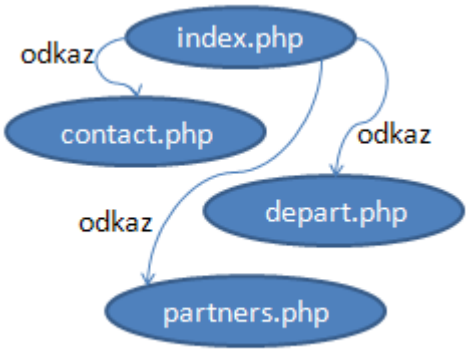
1. na začiatku vyhľadávač nájde stránky zdravotných stredísk
   - resp. stránky, ktoré majú príznak, že prezentujú konkrétne zdravotné stredisko
2. z hlavnej stránky vedie odkaz s príznakom *„oddelenia"* na zoznam oddelení
   - na stránke za týmto odkazom nájde PC *„<ul>"*[28] predstavujúci zoznam, ktorý bude mať dodatočné informácie, že predstavuje *„zoznam oddelení"*. PC prehľadá

---

[26] Enterprise Service Bus: http://en.wikipedia.org/wiki/Enterprise_service_bus
[27] Web Crawler: http://en.wikipedia.org/wiki/Web_crawler

[28] HTML značka pre nezoradený zoznam: http://www.w3schools.com/tags/tag_ul.asp

Obrázok 6: Väčšina súčasných webov



Obrázok 5: Ako by to mohli vyzerať

zoznam a nájde/nenájde prvok s príznakom *„ortopedické oddelenie"*, pričom si poradí aj s napr. Chorvátskym prekladom *„ortopedski"*

3. s odkazom z hlavnej stránky s príznakom *„akceptované cestovného poistenia"* si PC poradí obdobne ako so zoznamom oddelení
4. na stránke za odkazom s príznakom *„kontakt"* nájde PC kontaktné údaje, medzi ktorými budú/nebudú *GPS*[29] súradnice

Ale ako by PC mal vedieť, že za odkazom s príznakom *„kontakt"* nájde adresu, ktorej súčasťou by mali byť aj *GPS* súradnice? A ako by mal vedieť, že čo údaj *GPS* znamená, a že pomocou neho je možné určiť polohu. Tu prichádzajú na scénu *Ontológie*. Slovo ontológia ma značne široký a multidisciplinárny význam. Zaoberajú sa ňou obory ako filozofia, umelá inteligencia, systémové inžinierstvo, informačný manažment, výpočetná lingvistika a kognitívna psychológia. Okrem primárneho významu ako filozofickej disciplíny o *„učení o bytí"*[30], je pre naše účely najdôležitejší význam ako :

„...výslovný (explicitný) a formalizovaný popis sveta, alebo niektorej jeho časti, určitej problematiky. Je to formálna a deklaratívna reprezentácia, ktorá obsahuje slovník pojmov (definíciu pojmov) a vzťahy medzi týmito pojmami"

Ontológia je teda to, čo PC napovie, čo by sa malo nachádzať na stránke, ktorej odkaz má príznak *„kontakt"*, alebo čo znamená nájdený údaj *GPS* a čo sa s týmto údajom dá robiť (napr. určiť vzdialenosť od našej aktuálnej pozície). Prefixy *„p:"* a *„f:"* z **Error! Reference source not found.** by mohli predstavovať ontológie kde

[29] GPS: http://en.wikipedia.org/wiki/Global_Positioning_System
[30] Viac o ontológiách: http://cs.wikipedia.org/wiki/Ontologie

- „f:" by mohla predstavovať ontológiu základných sociálnych pojmov[31]
- „p:" zasa ontológiu so zdravotnou problematikou[32]

Pred tým, ako si predstavíme bližšie riešenie sémantického webu v kapitole 4 Štruktúra riešenia sémantického webu, povieme si niečo o vývoji samotných ontológií v kapitole 3 Ontológie.

[31] Napríklad FOAF: http://www.foaf-project.org/original-intro alebo SUMO: http://www.ontologyportal.org

[32] Napríklad SNOMED CT: http://en.wikipedia.org/wiki/SNOMED_CT

3. Ontológie

Ako už bolo spomenuté, pojem ontológia je multidisciplinárny. V tejto práci sa budem zameriavať na ontológiu používanú v obore *reprezentácie znalostí*[33] (ďalej ako RZ).

RZ je súčasťou oboru *umelej inteligencie*[34] (ďalej ako UI), ktorý sa snaží vyriešiť problémy spojené s návrhom a použitím formálnych jazykov vhodných pre zachytenie ľudských znalostí. Cieľom tejto formálnej reprezentácie je umožniť inteligentným systémom automatizované odôvodňovanie na základe tejto znalosti, čo týmto systémom umožní vykonávať úlohy normálne vykonávané človekom. Inteligentný systém musí uchovávať internú reprezentáciu tej časti reality, v ktorej má prejavovať inteligenciu, teda potrebuje pochopiť svoju doménu. Toto pochopenie je všeobecne zachycované v tzv. modeloch, na základe ktorých je počítač schopný prevádzať automatizované odôvodňovanie. Napríklad model o ľudskom tebe by mal mimo iné stroju umožniť zistiť, že ("zdravé") telo má 2 ruky, na nich po 5 prstov… . Modely nie sú priamo reflexiou reality, ale skôr našich znalostí o danej doméne. Konštrukcia takýchto modelov preto zahŕňa mapovanie medzi ľudským porozumením a porozumením počítača. Mapovanie ľudskej znalosti do počítačových modelov je najťažší problém oboru UI. Jedným z typov týchto modelov sú aj *ontológie*.

Znalostné inžinierstvo[35] (ďalej ako ZI), analógia *softwarového inžinierstva*[36], je obor ktorý sa zaoberá špecifikáciou a návrhom takýchto systémov. ZI je dôležitým aspektom *získavania znalostí*[37] (ďalej ako ZZ), všeobecný problém toho ako znalosť od ľudských expertov získať a zorganizovať aby to bolo implementovateľné a znovapoužitelné. Výsledkom snáh ZI sú početné metodológie a princípy návrhu pre tvorbu ontológií. *Web Ontology Language* (víz. kapitolu o štruktúre riešenia sémantického webu) je dôležitý člen rodiny RZ jazykov, navrhnutý špeciálne pre ontológie. Vývoj ontológií v tomto jazyku je považovaný za zložitý, čo vedie k vývoju nástrojov, zlepšovaniu metodológií, špecifikácií ontologických návrhových vzorov ako aj k vývoju niekoľkých ontológií, ktoré by mali slúžiť ako *framework*[38] (ďalej ako FW) pre viaceré domény[39].

---

[33] Knowledge representation: http://en.wikipedia.org/wiki/Knowledge_representation_and_reasoning

[34] Artificial Intelligence: http://en.wikipedia.org/wiki/Artificial_intelligence

[35] Knowledge engineering: http://en.wikipedia.org/wiki/Knowledge_engineering

[36] Software engineering: http://en.wikipedia.org/wiki/Software_engineering

[37] Knowledge acquisition: http://en.wikipedia.org/wiki/Knowledge_acquisition

[38] Všeobecná zjednocujúca základná kostra, viz napr: http://en.wikipedia.org/wiki/Software_framework

3.1.Vývoj pojmu ontológia v informačných technológiách

Pojem ontológia má veľa významov. Veľké množstvo publikácií sa točí okolo nejakej konkrétnej ontológií, napríklad o nejakom ontologickom jazyku, metodológií na tvorbu ontológií alebo konkrétnych typoch ontológií, pričom každá z týchto publikácií obsahuje nejakú definíciou toho, čo ontológia je. Najčastejšie citovaná je asi definícia :

„Ontológia je explicitná špecifikácia konceptualizácie“[40]

Existuje nespočetne alternatívnych definícií ontológie, ktoré sú v prípade vybratia z kontextu rovnako neinformatívne. Tieto ontológie môžu predstavovať od ľahkého textového popisu výrazu až po vysoko formálnu špecifikáciu filozofických primitív. Prvé použitie pojmu ontológia ako popisu všetkých znalosti bolo v (1). Následne bolo adoptované komunitou okolo ZZ, ako pojem pre stavebné kamene doménovej teórie, konceptov. Neskôr sa začal používať ako pojem v špecifikácií *systémov založených na znalostnej báze* [41] (ďalej ako SZB) pre špecifikáciu doménovej znalosti vo forme dokumentácie, schematických diagramov alebo textových popisov. Medzi ďalšie z použití patrí napr. znalostná komponenta implementujúca odôvodňovaciu službu alebo poskytujúca kostru pre ZZ v nejakej doméne a postupne rozširované o znalosti z domény. ZZ pohľad nadobúdal významu potom ako zistilo, že môže slúžiť aj ako prostriedok pre komunikáciu v celom systéme, alebo aj medzi systémami a dokonca aj medzi systémom a ľuďmi. Nasledovalo použitie pre zdieľanie znalostí medzi skupinami ľudí/v rámci organizácie, zachycovanie cieľov organizácií a podnikových procesov : *znalostný manažment*[42] v organizáciách. Časom, kvôli silnej schopnosti vylepšovať kooperáciu medzi organizáciami, začal slúžiť aj ako slovník všeobecne prijatých pojmov, teda za účelom štandardizácie. Rozšírením internetu vzrastal záujem o potrebou vývoja jazyka na výmenu informácií.

3.2.Ontologické inžinierstvo

V priebehu rokov bolo zo softwarového inžinierstva do ontologického inžinierstva prevedené niekoľko princípov návrhu kvôli snahe o zlepšenie kvality vývoja ontológií. Na šťastie na internete je rozšírený dostatok všeobecných ontológií, ktoré poskytnutím počiatočnej štruktúry a základnej množiny konceptov a vzťahov môžu predstavovať cenný

---

[39] Existuje niekoľko ontológií, ktoré je možné považovať za vzorové pre vývoj ontológie. Poskytujú pohľad ako na kvalitu a návrh, tak na kompromis medzi teóriou a praxou

[40] *„An ontology is an explicit specification of a conceptualization“* (5)

[41] Knowledge-based systems: http://en.wikipedia.org/wiki/Knowledge-based_systems

[42] Knowledge Management: http://en.wikipedia.org/wiki/Knowledge_management

začiatok pre vývoj špecifickejšej ontológie. Medzi najnovšie výsledky vývoja v ontologickom inžinierstve patrí identifikácia návrhových vzorov, ktoré sa snažia prekonať niektoré obmedzenia v znovapoužití existujúcich ontológií (ktoré vedia byť veľké, ťažkopádne a ťažko formovateľné a rozšíriteľné na použiteľnú doménovú ontológiu) a aplikácií návrhových princípov (abstraktné a ťažko preložiteľné na definíciu konkrétnej ontológie). Vzory sú akýmsi medzistupňom oboch.

3.2.1. Metodológie a kritéria

Zo skúseností vo vývoji veľkých ontológií v 90.tych rokoch vzišla potreba metodologického základu. Vzniklo niekoľko publikácií so skúsenosťami s vývojom ontológií. V jednej z nich (skúsenosti z vývoja ontológie *Ontolingua systém*[43] (2)) bola identifikovaná potreba správneho pomeru medzi 5 kritériami návrhu (3):

1. ***Jasnosť*** – efektívne definovanie výrazu, teda definícia by mala byť objektívna, formálna podľa možností, dokumentovaná prirodzeným jazykom
2. ***Súdržnosť/súvislosť*** – ontológia by mala povoľovať iba vyvodenia, ktoré sú konzistentné s definíciou
3. ***Rozšíriteľnosť*** – štruktúra by mala povoľovať rozšírenie bez úpravy samotnej ontológie
4. ***Minimálne skreslenie kódovaním*** – konceptualizácie by mala byť špecifikovaná na znalostnej úrovni, ktorá nie je závislá na konkrétnych symboloch určitého kódovania
5. ***Minimálny ontologický záväzok*** – ktorý je potrebný pre podporu zamýšľaných aktivít

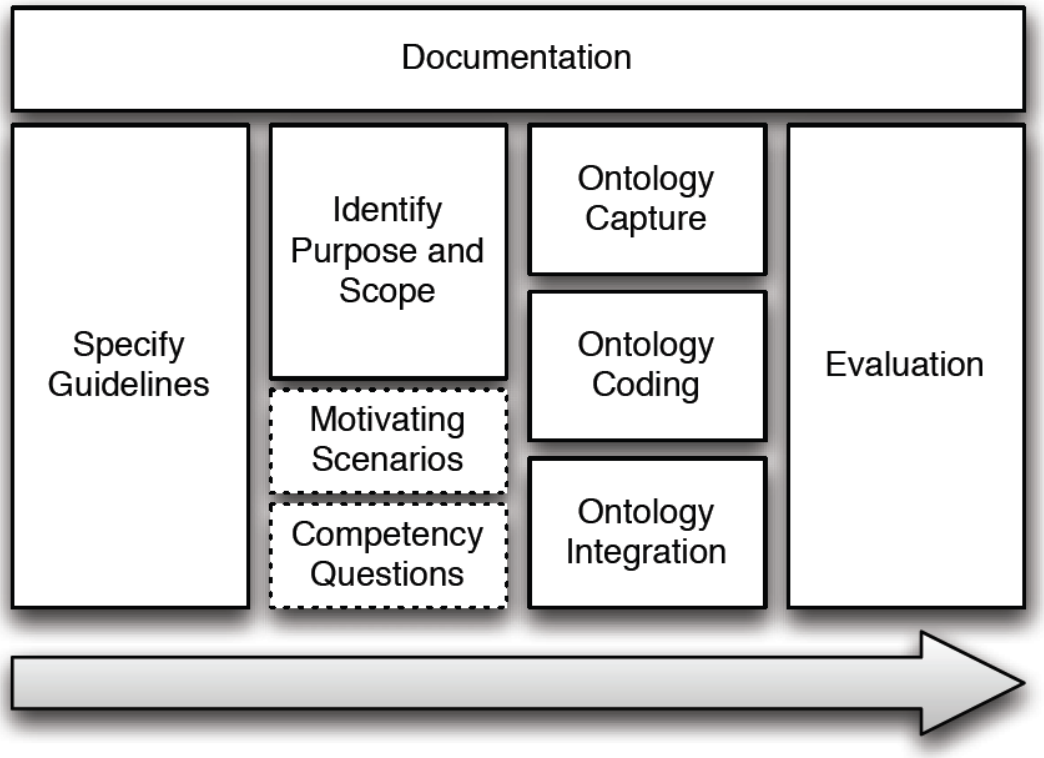
O týchto kritériách nemožno uvažovať samostatne a príchod OWL ako štandardu ich do istej miery zafixoval. Kombináciou skúseností z vývoja ontológií *TOVE*[44] a *Enterprise ontologies*[45] vznikla kostrová metodika (4), ktorá nasleduje princípy vývoja ontológií stanovených pánom Gruberom (5) a viacero metodológii vychádza práve z nej. Metodológia sa rozdeľuje vývoj do niekoľkých fáz (víz. Obrázok 7) :

1) ***Stanovenie smerníc*** – ktoré ovládajú celý vývoj, od jednoduchých ako pomenovávacie konvencie po hlasovací systém v prípade zmeny.

---

[43] Ontolingua: http://www.ksl.stanford.edu/software/ontolingua/
[44] TOVE: http://www.eil.utoronto.ca/enterprise-modelling/tove/
[45] Enterprise ontologies: http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html

Obrázok 7:Fáze metodológie (4)
Zdroj: (12 p. 84)



2) ***Dokumentácia*** – ktorá by mala sprevádzať celý proces vývoja a ktorá by mala zachytávať mimo iné aj všetky rozhodnutie ohľadom návrhu ontológie
3) ***Identifikácia účelu a rozsahu*** – jasná informácia o použití a užívateľoch ontológie značne pomáha pri prípadných návrhových rozhodnutiach.
    a) Motivačné scenáre a z nich odvodené otázky na kompetenciu – dopomáhajú k identifikácií použitia ontológie.
4) ***Samotná tvorba ontológie*** – ktorá sa skladá z nasledujú paralelne bežiacich a blízkych procesov
    a) *Zachytenie* – identifikácia kľúčových konceptov a vzťahov, vytvorenie presných definícií v prirodzenom jazyku a výber vhodných výrazov na odkazovanie s na daný element
    b) *Kódovanie* – teda reprezentácia primitív v nejakom jazyku, čo so sebou zahŕňa záväzok ku konkrétnemu znalostnému formalizmu. Ten má vplyv ako na samotnú štruktúru ontológie, tak na interpretáciu. Úroveň formalizmu je potreba voliť vhodne podľa typu a účelu ontológie.
    c) *Integrácia* – teda využitie znovapoužiteľnosti iných ontológií, ktoré musí byť riadne zvážené, teda z pohľadu účelu vytváranej a importovanej ontológie, konceptov ktoré nie sú pre vytváranú relevantné.

5) ***Evaluácia*** – finálna fáza vývoja, kedy sa ontológia vyhodnocuje voči spočiatku stanoveným požiadavkám a prípadná vhodnosť zistených odchyliek

3.2.2. Proces štandardizácie

Predpoklad o znovapoužívaní a zdieľaní ontológií so sebou niesol očakávanie, že ontológie budú veľké a teda vyvíjané veľkým množstvom ľudí. Tento predpoklad priniesol potrebu existencie procesu štandardizácie. Skutočnosť je ale iná. Väčšina súčasných a najviac používaných, ako veľkých tak malých, ontológií ako FOAF[46], *Dublin Core*[47], *RSS*[48], *LKIF*[49] bolo vyvinutých malým množstvo ľudí. Ontológie, ktoré sú výsledkom vývoja väčším množstvom ľudí, sú hlavne vedeckého rázu, ako napríklad SNOMED (ontológia medicínskych pojmov) alebo *SUO*[50]. Vývoj SUO, čo bolo iniciatíva *IEEE*[51], nakoniec skončil na neschopnosti komunity dohodnúť sa na niekoľkých „*horúcich*“ témach. Do ďalšieho vývoja ontológie Dublin Core, ktorá pôvodne vznikla spoluprácou malej skupinky ľudí, a ktorá si získala značné množstvo užívateľov, sa chcelo zapojiť väčšie množstvo komunít, ktoré mali v ontológií svoje zaujmi, čo viedlo k podobnej pätovej situácií ako u vývoja SUO.

3.2.3. Vytváranie ontológie

Medzi majoritné používané spôsoby samotného vytvárania konceptov a vzťahov patria prístupy *„z hora dolu“* [52]alebo opak *„z dolu hore“*[53]. Začatie „*hore*“ so sebou nesie potencionálne problém nepresnosti teda nebezpečenstvo, že špecifickejšie koncepty nebudú sedieť na všeobecnejšie čo môže viesť k nekonečným úpravám vrchných konceptov. Opačný smer hrozí vytvorením veľkého množstva nepotrebných a detailných konceptov. Ako kompromis by sa dal nazvať prístup navrhnutý pánom Lakoffom (6), ktorý navrhuje vytvorenie tzv. skupiny konceptov *„základného stupňa“*[54], ktoré sú ľuďmi podvedome najčastejšie používané a pomocou ktorých je možne popísať ostatné koncepty. Túto teóriu podporuje aj výskum pána Ogden, *Basic English* [55] (7), ktorý ukázal, že 90% anglického slovníka môže byť rozumne popísaný použitím zvyšných 10% slov (nejakých 840 slov). Jedná sa o slová ako *pes*, *kladivo*, *stôl*. Pomocou týchto konceptov je možné pomerne

---

[46] FOAF: http://www.foaf-project.org/
[47] Dublin Core: http://dublincore.org/
[48] RSS: http://en.wikipedia.org/wiki/RSS
[49] LKIF: http://www.estrellaproject.org/lkif-core/
[50] SUO: http://suo.ieee.org/
[51] IEEES: http://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers
[52] Top-down & bottom-up design: http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design
[53] Top-down & bottom-up design: http://en.wikipedia.org/wiki/Top-down_and_bottom-up_design
[54] Basic Level
[55] Basic English: http://en.wikipedia.org/wiki/Basic_English

jednoducho popisovať koncepty ako smerom hore, teda všeobecnejšie ako slová *cicavec* alebo *nábytok*, tak smerom dole, napr. *bradáč*, *hydraulické kladivo*. Aj táto metóda má žiaľ problém v tom, že v OWL tvorí definíciu konceptu jeho umiestnenie medzi ostatnými konceptmi, ktoré ešte neexistujú, ako aj to, že čím základnejšie koncepty, tým zložitejšie je vymyslieť rozumnú definíciu daného konceptu z pozície medzi ostatným konceptmi.

3.2.4. Integrácia a znovapoužitie

Ontológie slúžia na uľahčenie získavania znalostí poskytnutím použiteľných definícií konceptov. Vývoj ontológií má podobný účel, existujúce ontológie môžu slúžiť ako použiteľný zdroj pre tvorbu ontológii vo forme preddefinovaných konceptov a vzťahov, teda poskytnúť iniciálnu štruktúru. Tento spôsob vývoja môžeme ešte rozdeliť na integráciu (kombinácia dvoch a viac ontológií do jednej výslednej) a znovapoužitie (ontológie sú udržiavané oddelene), pričom oba spôsoby so sebou prinášajú niekoľko problémov. Použité ontológie musia byť *zarovnané*, teda sémanticky blízke koncepty a vzťahy by mali byť spojené vzťahom ekvivalencie resp. podmnožinou. V prípade väčšieho presahu ontológií to znamená významné zmeny v pôvodných ontológiách. Zmeny sú potrebné ak kombináciou ontológií vzniká heterogénny systém v ktorom existujú nezhody na jazykovej (v spôsobe definovania konceptov a relácií), ontologickej alebo sémantickej úrovni (v predpokladoch o doméne).

Jazykové nezhody môžu predstavovať

- syntaktické nezhody ako je nekompatibilita syntaxe jazyka *KRSS*[56] a *RDF/XML*[57]
- viac fundamentálne problémy spôsobené vyjadrovacími možnosťami
    - celkovo alebo iba v niektorých špecifických častiach jazykov.

Prevod z jazyka menej do viac vyjadrovacieho vedie k strate informácií, prevod opačný je menej problematický, ale vyžaduje mapovanie a prepisovanie fráz.

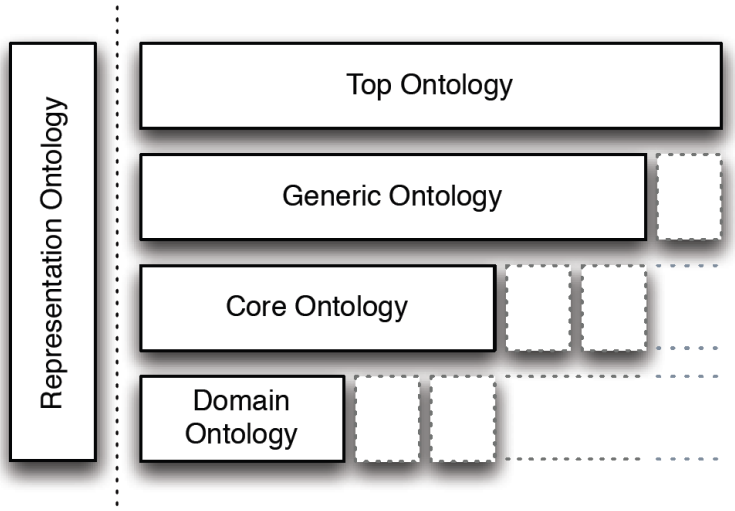
Sémantické nezhody sú častejšie. Rôznosť samotného ontologického prístupu (vnímania sveta) napr.

- zaznamenávanie času ako intervalov verzus bodové uchovávanie času
- uchovávanie hodnôt ako konštánt verzus dátových typov

[56] KRSS: http://dl.kr.org/

[57] RDF/XML: http://www.w3.org/TR/rdf-syntax-grammar/

Obrázok 8: Typy ontológií podľa (9)
Zdroj: (12 s. 94)



Integrácia existujúcich ontológií je zložitejšia ako priame znovapoužitie existujúcej, ale dáva nám dobrý náhľad nato čo je potreba si postrážiť pri výbere ontológie na znovapoužitie: pokrytie domény, stupeň abstrakcie ontológie, granualita, reprezentačný jazyk a stupeň formálnosti.

Znovapoužitie ontológií bolo horúce téma hlavne v dobe konštrukcie ontologickej knižnice (8) (9), počas ktorého pán Heijst rozlíšil 5 typov ontológií (víz. Obrázok 8) podľa stupňa abstrakcie ich obsahu, ktorého hodnota mala mať vplyv na možnosti znovapoužiteľnosť ontológie :

1. Top – silno všeobecné kategórie a vzťahy relevantné pre všetky domény
2. Všeobecná (*Generic*) – všeobecné pojmy aplikovateľné cez viacero domén, ako napríklad čas, priestor atď.
3. Jadrová (*Core*)- všeobecné typy ale už pre konkrétnu doménu, ako napríklad právnictvo, medicína, pomocou ktorý je možné popísať entity v danej doméne
4. Doménová (*Domain*) – koncepty konkrétnej domény
5. Aplikačná (*Representation*) – zachycuje konkrétne entity pre konkrétnu aplikáciu, ktorou je používaná

Toto rozdelenie je umelé a v praxi sa iba zriedka dá o nejakej konkrétnej ontológií povedať, že by patrila iba do jednej z kategórií. Súčasné ontológie sa dajú rozdeliť do kategórií, ktoré sú kombináciou vlastností spomenutých typov :

- **Zjednotené** (*Unified*) – obsahujúca silne abstraktné tak ako konkrétne koncepty. Najtypickejší predstaviteľ *SENSUS*[58] bol vytvorený priamo s úmyslom znovapoužitia spolu s metodológiou používajúcou túto ontológiu v jednom z krokov vývoja. V konečnom dôsledku tieto ontológie nie sú vhodné pre znovapoužitie
- **Podkladové** (*Foundational*) – sú kombináciou Top ontológií s úrovňou popisu aplikačných ontológií. Majú filozofickú dispozíciu a sústredia sa na viac abstraktné koncepty, základné kategórie existencie a majú široký záber zachádzajúci do veľkého množstva domén. Aj napriek neprítomnosti konkrétnejších konceptov je vzdialenosť medzi konceptmi takejto ontológie stále veľká a jej znovapoužiteľnosť v skutočných ontológiách je otázna.
- **Jadrové** (*Core*) – obmedzujú svoj rozsah nad konkrétnou oblasťou. Aj napriek malému rozsahu, sú vhodnejšími kandidátmi na znovapoužitie, kvôli množstvu konceptov skutočne použiteľných v prípade znovapoužitie v ontológií nejakej domény.
- **Doménové** (*Domain*) – rozširujú jadrové o konkrétnejšie koncepty.

Obmedzenie vyjadrovacej sily znovapoužívanej ontológie je niekedy nevyhnutné. V praxi je to často viditeľné pri aplikácií podkladových ontológií. Napríklad ontológia *SWIntO*[59] (10), ktorá je kombináciou ontológií SUMO a DOLCE, pričom z ontológie SUMO boli vynechané skoro všetky axiómy (ostala iba hierarchická kostra podtried). Tento proces ale ochudobňuje ontológiu ako zdroj rozhodnutí, návrhových vzorov a účelu ontológie. Ani importom všetkých axiómou kompletnej ontológie nemáme boj so znovapoužívaním vyhratý. Príkladom môže byť :

- Ontológia B je vytváraná znovapoužitím ontológie A
- Nech A tvrdí
    - *Vták má presne 2 krídla*
    - *Tučniak je vták*
- V A teda platí : *Tučniak má presne 2 krídla*
- Nech B tvrdí : *Vták vie lietať*
- Importom A bude platiť : *Tučniak vie lietať*

[58] SENSUS: http://www.isi.edu/natural-language/projects/ONTOLOGIES.html

[59] SWIntO: http://smartweb.dfki.de/ontology_en.html

Zjednodušená definícia (11) vraví, že znovapoužitie je bezpečné ak: *„nie sú implikované žiadne axiómy v ontológii, ktorá importuje, nad symbolmi z ontológie, ktorá je importovaná, ktoré nie sú implikované ani v importovanej ontológií.“*

3.2.5. Frameworky (OntoClean)

Medzi ďalšie pomôcky pri vývoji patria rôzne FW ako je napr. *OntoClean*[60], ktorý poskytuje prostriedky a metodológiu na vyhodnotenie správnosti vzťahov medzi entitami. Centrálne typy overovaných vzťahov sú :

- **Podstata** (*Essence*) - členstvo v triede je pre nejaký entitu podstatné, teda entita nemohla existovať ak by nepatrila do daného typu triedy
- **Pevnosť** (*Rigid*) - trieda je pevná ak všetky entity, ktoré sú členom tejto triedy, sú existenčne závislé na členstve v tejto triede.
- **Identita** (*Identity*)– ako určiť, či 2 entity sú to isté
- **Jednota** (*Unity*) - celok zložený z častí

Medzi ďalšie typy FW patria napríklad *Situačné FW*, ktoré pomáhajú poskytnutím stereotypných štruktúr používaných pre dosiahnutie nejakého cieľa (napríklad *zabitie klinca do steny*) a *Mereologické FW*, zamerané na popis entít pomocou popisu všetkých častí entity (napríklad *auto, ktoré má minimálne 3 ale zvyčajne 4 kolesá a jeden motor*) a iné..

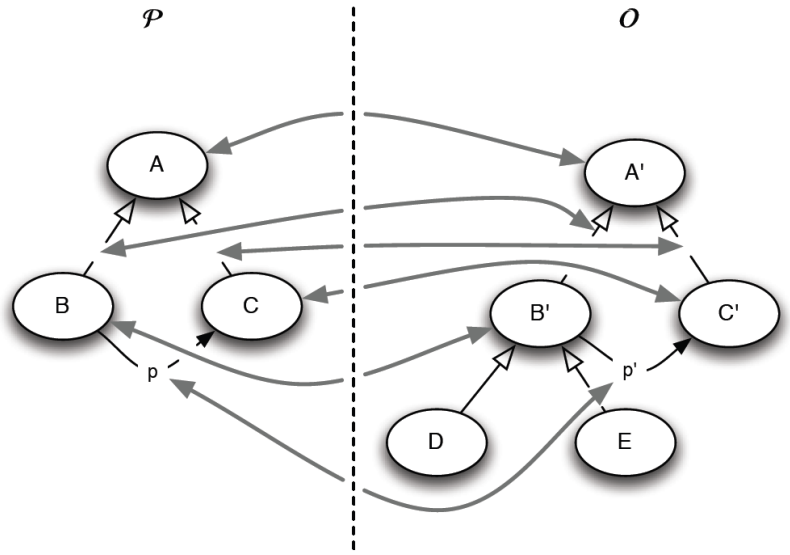
3.2.6. Návrhové vzory

Doterajšie postupy a pomôcky môžeme nazvať ako *znalostne modelovacie*, sú konceptuálne a nezávislé na použitom jazyku. Výber reprezentačného jazyka a teda hlavne jeho vyjadrovacích možností určuje, ktoré ontológie je možné znovapoužiť. Ako pri znalostnom modelovaní bolo potreba spraviť rozhodnutia o zvolenom riešení, tak pri samotnom návrhu ontológie je potreba robiť podobné rozhodnutia. Viacerými ontológiami overené rozhodnutia sa môžu považovať za tzv. *návrhové vzory (Ontology Design Pattern)* (ďalej ako ODP). Za návrhový vzor považujeme *typické riešenie návrhového problému v určitom kontexte* (12 s. 111).

Ontologický popis konceptov vo formálnom jazyku sa veľmi nelíši od popisu výrazov v prirodzenom jazyku. V oboch prípadoch je popis vytvorený kombináciou symbolov podľa nejakého gramatického pravidla. Rola návrhových vzorov v ontológiách by sa dala prirovnať k gramatickým pravidlám v prirodzenom jazyku. V skutočnosti náš spôsob kombinácie slov v lingvistické výrazy poukazuje na existenciu niekoľkých základných konceptov

[60] OntoClean: http://en.wikipedia.org/wiki/OntoClean

Obrázok 9: Mapovanie medzi ODP *P* a jeho implementáciou v ontológií *O*
Zdroj: (12 s. 115)



v myšlienkach a nasledovaním základných kognitívnych pravidiel. *Znalostný vzor* (*Knowledge pattern*) je definovaný *„ako teória, ktorej axiómy nie sú súčasťou znalostnej báze, ktorá ho implementuje“* (13). Implementácia vzoru sa prevádza importovaním axiómou vzoru a mapovaním symbolov implementácie na symboly vzoru. Mapovanie je dané *morfizmom*[61], ktoré mapuje každý nelogický symbol vo vzore do odpovedajúceho symbolu v znalostnej báze (viz. Obrázok 9) (viz. Tabuľka 2).

Ontologická verzia znalostného vzoru, teda *ontologický návrhový vzor* (ďalej ako ODP) by bol definovaný ako : *„ODP je ontológia P, o ktorej sa povie že je implementovaná v ontológii O, ak existuje funkcia M špecifikovaná morfizmom, ktorá mapuje medzi signatúrou P a O a pre každý axióm a z P platí, že O |= M(Sig(a),a), kde M(Sig(a),a) označuje množinu axiómou vzniknutých aplikovaním M na jeho signatúru.“* (12 s. 115). ODP môžeme ešte deliť na:

- **obsahové** – sú v podstate malé ontológie, ktorých znovapoužitie je riadené dokumentáciou ich logickými odôvodneniami a *best-pracices*. Sú definované ako obmedzenia na znalostných vzoroch.
- **logické** – sú špecifikované na *meta-úrovni* a sú implementované inštanciáciou. Axiómy v implementácií sú typované podľa *meta-kategórií* definovanými vzorom. Definujú *meta-triedy* a skonkrétnené vzťahy, ktoré špecifikujú platnú implementáciu vzoru. Príkladom sú *top* úrovňové kategórie samotného OWL, ktoré umožňujú definovanie ľubovoľnej platnej OWL ontológie.

[61] Morphism: http://en.wikipedia.org/wiki/Morphism

- **Štrukturálne** - veľa vzorov nie je často definovaných ako formálne teórie, ale skôr ako procedúry s typickými ukážkami ako krok po kroku dosiahnuť napr. hlavne aproximácie štruktúr, ktoré nie je možné priamo vyjadriť v OWL(napr. N-árne relácie).

Čo	Príklad
Vzor	*Voľná_pamäť ⊆ Kapacita ⊓ kapacita **nejakého** Kontainer ⊓ **nie** Použitá* *Kontainer ⊆ voľná_pamäť **nejakej** Voľnej_pamäte* *voľná_pamäť ⊆ kapacita_niečoho⁼*
Morfizmus	*m = {(Kontainer, Počítač), (Voľná_pamäť, Dostupná_RAM),* *(Kapacita, Veľkosť_RAM), (Použitá, Použitá_RAM),* *(voľná_pamäť, dostupná_ram)}*
Implementácia	*Dostupná_RAM ⊆ Veľkosť_RAM ⊓ kapacita_niečoho **nejakého** Počítača* *⊓ **nie** Použitá_RAM* *Počítač dostupná ram nejakej Dostupnej RAM* *dostupná ram má kapacitu*

Tabuľka 2: Príklad mapovania ODP *Voľná pamäť* na *Dostupnú RAM*
Zdroj: **(12 s. 114)**

3.2.6.1.1. Diamantové schéma

Ako príklad návrhového vzoru uvedieme často vyskytujúci sa jav, transakciu, ktorá spája dve akcie identitou a obmedzeniami na rolách zúčastnených akcií, a ktorá takto vytvára graf v tvare diamantu. Podstatou transakcie je výmena/zmena rôzneho druhu (fyzická, psychická, ...). Je to nevyhnutná konštrukcia vyskytujúca sa vo veľa podobách. Napríklad *zmena farby svetla na semafore*, *roztavenie pevnej látky*. Takáto zmena so sebou nesie obmedzenia na vzájomnosti, napríklad *obchodná transakcia dvoch partnerov*: *kupujúci dostáva tovar za ktorý platí peniaze a predávajúci dostáva tie isté peniaze za ktoré dodáva ten istý tovar*. Platí teda :

- Identita – aktívny zúčastnení v oboch akciách sú rovnaký (*predávajúci, kupujúci*)
- Vyvážanie – objekty v oboch akciách by mali byť porovnateľné (*peniaze vyvážiť cenu tovaru*)

Vytvorenie tohto vzoru by malo nasledovať tieto kroky (12 s. 152-159):

1. Počiatočné zadefinovanie triedy – typu *transakcia*, ktorá má časti akcie *prevodu*. Tá má pre zmenu vlastnosti *prijímateľa* a *dodávateľa* s hodnotami typu *Agent* a vlastnosť *objekt* s typom *Objekt*.

```
Transakcia   ≡     má_časť nejaký Prevod
Prevod       ≡     dodávateľ nejaký Agent
                ∩  prijímateľ nejaký Agent
                ∩  objekt nejaký Objekt
```

2. Obmedzenia množstva – *prevodov* a ich ne-identity

```
Transakcia   ≡     má_časť nejaký Prevod
                ∩  má_časť min 2 Prevod
             ⊆     má_časť iba Prevod
                ∩  má_časť presne 2 Prevod
```

3. Obmedzenie množstva – v *prevode*

```
Prevod     ≡     prijímateľ nejaký Agent
              ∩  dodávateľ nejaký Agent
              ∩  objekt nejaký Objekt
           ⊆     prijímateľ presne 1 Agent
              ∩  dodávateľ presne 1 Agent
              ∩  objekt presne 1 Objekt
```

V prípade chcenia rozlíšenia *prijímateľa* a *dodávateľa* je možné pridaním vlastnosti disjunkcie.

4. Porovnateľnosť hodnôt *objektov* – z aktuálne podoby grafu (víz. **Error! Reference source not found.**) a pomocou zápisu :

```
objekt o má_časť⁻ o má_časť o objekt ⊆ hodnota_podobná
```

5. Asymetria stromu – ktorú je možné dosiahnuť spomínanou vlastnosťou disjunkcie, alebo rozlíšením strán stromu grafu pomocou rozlíšenia povâh zúčastnených a to ich charakterizáciou. Dajme tomu, že ide o predaj *tovaru* za *peniaze* (prevod *peňazí* sa doplní obdobne), tak :

Tovar_Prevod	≡	Prevod
	∩	objekt **nejaký** Tovar
prijímateľ$_g$	⊆	Prijímateľ
dodávateľ$_g$	⊆	Dodávateľ
Tovar_Prevod	≡	prijímateľ$_g$ **nejaký** Agent
	∩	dodávateľ$_g$ **nejaký** Agent

(víz. Obrázok 10)Vďaka presným uvedeným kardinalitám na *Prevode* pre *prijímateľa*, *dodávateľa* a *objekt*, by DL odôvodňovač mal odvoďiť, že pre :

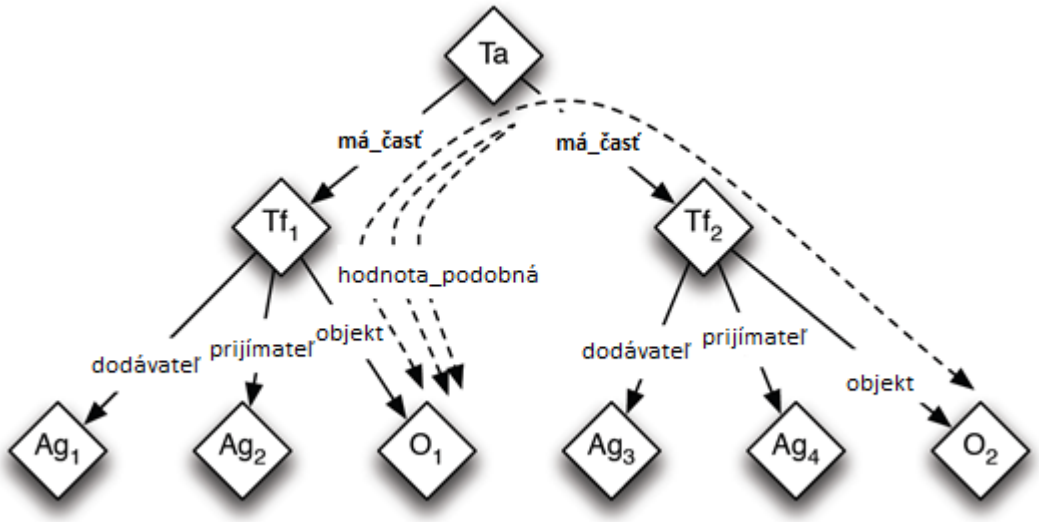
ľubovoľný Tovar_Prevod ***g*** *a Agent* ***a****, ak* ***g*** *dodávateľ* ***a*** *potom* ***g*** *dodávateľ*$_g$ ***a****.*

Dodaním vzťahov s vlastnosťou *rovnaké_id_ako* v kontexte špecifických relácií

Dodávateľ$_g^-$ **o má_časť$^-$ o má_časť o** prijímateľ$_m$	⊆	rovnaké_id_ako
Dodávateľ$_m^-$ **o má_časť$^-$ o má_časť o** prijímateľ$_g$	⊆	rovnaké_id_ako
Prijímateľ$_g^-$ **o má_časť$^-$ o má_časť o** dodávateľ$_m$	⊆	rovnaké_id_ako
Prijímateľ$_m^-$ **o má_časť$^-$ o má_časť o** dodávateľ$_g$	⊆	rovnaké_id_ako

by na transakcii zachytenej na obrázku 4 a dodaním : O_1 *: peniaze,* O_2 *: Tovar* môžeme odvodiť, že: Ag_1***rovnaké_id_ako*** Ag_4 *a* Ag_2 ***rovnaké_id_ako*** Ag_3 .

Obrázok 10: Strom vznikajúci krokmi pre tvorbu vzoru transakcie
Zdroj: (12 s. 156)



4. Štruktúra riešenia sémantického webu

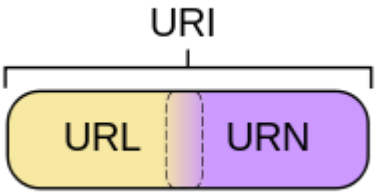
Koncept sémantických sietí pochádza zo začiatku 60.rokov 19.storočia ako forma významovo štruktúrovaných vedomostí. Ich zakladateľmi boli kognitívny vedec Allan M. Collins, lingvista M. Ross Quillian a psychologička Elizabeth F. Loftusas. Myšlienku sémantického, alebo významového webu prvý krát vyslovil Tim Berners-Lee, v spolupráci s James Hendler a Ora Lassila, v roku 2001 v článku v magazíne Scientific American. Sémantický web definovali ako :

„web dát, ktorý môže byť spracovávaní priamo a nepriamo strojmi“[62]

Sémantický web je nerušivé rozšírenie súčasného webu, ktoré pridáva rôzne vrstvy strojovo spracovateľnej sémantiky(víz. Obrázok 12). Nerušivé preto, že každá vrstva je vytvorená na druhej bez toho, aby tú pôvodnú nejako modifikovala.

Prvá vrstva je tvorená štandardom pre *kódovanie znakov*[63], *UNICODE*[64] a pre identifikáciu zdrojov (na webe), *URI*[65]. Kódovanie znakov je laicky povedané spôsob zaznamenávania znakov do elektronickej podoby, pričom *UNICODE* je v tomto smere štandard, ktorý momentálne umožňuje reprezentáciu asi väčšiny svetových systémov pre písanie.

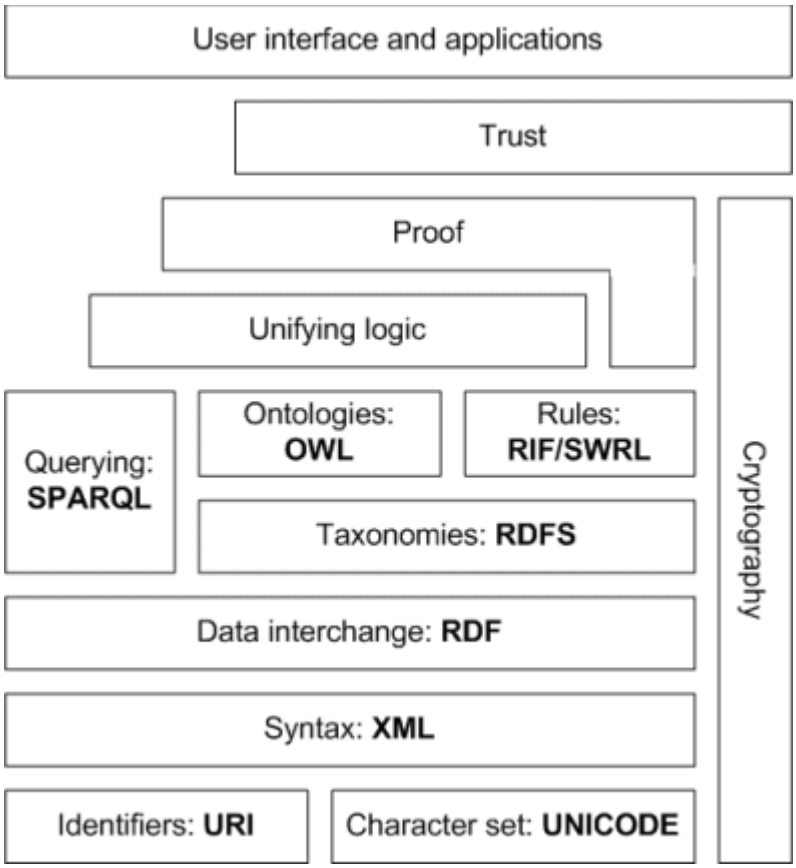
Sémantický web si pre identifikáciu adoptoval *URI*, čo je spôsob jedinečnej identifikácie zdrojov na (nie len) webe. Princíp je veľmi podobný dnešným *HTML* odkazom (ináč nazývaným aj *URL*[66]), pričom v skutočnosti každý odkaz je zároveň *URI*. Naopak to neplatí, pretože *URI* môže okrem *URL* časti obsahovať ešte *URN*[67] časť (víz. Obrázok 11).



Obrázok 11: Štruktúra URI
Zdroj:http://upload.wikimedia.org/wikipedia/commons/c/c3/URI_Euler_Diagram_no_lone_URIs.svg

[62] *„A web of data that can be processed directly and indirectly by machines."*
[63] Character encoding: http://en.wikipedia.org/wiki/Character_encoding
[64] UNICODE: http://en.wikipedia.org/wiki/Unicode
[65] Universal resource identifier: http://en.wikipedia.org/wiki/Uniform_resource_identifier
[66] Uniform resource locator: http://en.wikipedia.org/wiki/Uniform_resource_locator
[67] Uniform resource name: http://en.wikipedia.org/wiki/Uniform_Resource_Name

Obrázok 12: Zásobník sémantického riešenia
Zdroj: http://upload.wikimedia.org/wikipedia/en/3/37/Semantic-web-stack.png



Druhá vrstva je tvorená modulárnym mechanizmom pre popis štruktúry dát. *XML*[68] je značkový jazyk, ktorým sa okrem iného dajú definovať množiny pravidiel pre kódovanie (resp. štruktúru) dokumentov. Je ako ľudsky, tak strojovo čitateľný. Pod značkou si treba predstaviť dvojicu uzávorkovaných slov medzi ktorými sa môže nachádzať ďalší obsah, napríklad „`<test></test>`“. *XML* bol odvodený ako podmnožina *SGML*[69], čo je priemyselný *metadata*[70] (dáta o dátach) jazyk pre definovanie strojovo spracovateľných a výmenných formátov pre dokumenty. *XML* bolo prispôsobené pre účel výmeny dát na webe a pre kompatibilitu s jazykom pre tvorbu internetových stránok, *HTML*. Je na mieste upozorniť, že štruktúra vyjadrená pomocou *XML* nevyjadruje sémantiku dát. Doposiaľ spomenuté prostriedky SWS sú technológie vyvinuté pôvodne za iným účelom a sémantickým webom sú iba využívané na dosiahnutie určitého výsledku, preto v prípade záujmu čitateľa o ne odporúčam držať sa odkazov v poznámkach pod textom.

[68] eXtensible Markup Language: http://en.wikipedia.org/wiki/XML
[69] Standard Generalized Markup Language: http://en.wikipedia.org/wiki/Standard_Generalized_Markup_Language
[70] Metadata: http://en.wikipedia.org/wiki/Metadata

Prostriedky v stručnosti predstavené v tomto odstavci budú bližšie predstavené v nasledujúcich textoch. Tretia vrstva, vrstva *RDF* a *RDFS*, prináša prvú sémantiku. V podstate štandardom pre jednoduché *metadata* pre web sú jazyky *Resource Description Framework* [71] (ďalej ako RDF) a jeho slovníkové rozšírenie *RDF Schema*[72](ďalej ako RDFS), ktoré boli navrhnuté s prihliadnutím na spôsob uloženia informácií na webe. *OWL*, skratka pre *Web Ontology Language* [73], ktoré je súčasťou štvrtej vrstvy a rozširuje *RDFS* o pokročilejšie konštrukcie pre popis sémantiky *RDF* výrazov. Súčasťou štvrtej vrstvy je aj prostriedok na dotazovanie sa nad dátami spodných vrstiev, *SPARQL*. Skratka pre *SPARQL Protocol and RDF Query Language*[74].

Zvyšok prostriedkov SWS patrí medzi tzv. *nerealizované časti*. Ide o technológie, ktoré ešte nie sú štandardizované alebo dokonca obsahujú zatiaľ iba myšlienky, ktoré by mali byť implementované pre dokončenie celého riešenia sémantického webu.

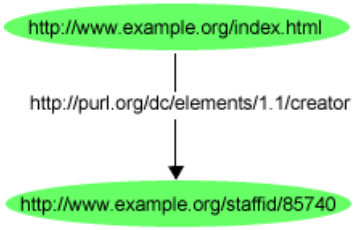
- *RFI*[75], *SWRL*[76] – prostriedky pre podporu pravidiel použiteľné napríklad na popis vzťahov, ktoré nie je možné popísať priamo v DL[77] v OWL
- Šifrovanie (Cryptography) [78] – nutné pre overenie, že tvrdenia prichádzajú z overených zdrojov
- Dôvera (Trust) – že odvodené znalosti pochádzajú z overených zdrojov a že ich správnosť sa opiera o formálnu logiku
- Užívateľské rozhranie (User interface)[79]- potrebné pre umožnenie využívania sémantických aplikácií ľudským užívateľom

4.1.RDF Resource Description Framework

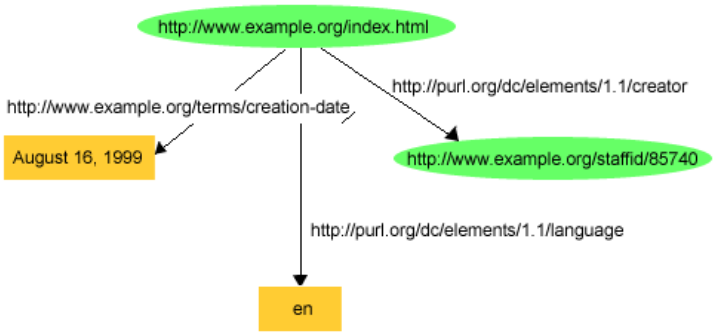
RDF je jednoduchý výrokový jazyk, ktorý definuje spôsob ako robiť tvrdenia o „*veciach*“ na (nie len) webe a tým ponúka ľahký a jednoduchý spôsob ako reprezentovať *metadata* na webe. Tieto tvrdenia sú robené vo forme trojíc, tzv. *triple:* „*<s, p, o>*“, syntaktická varianta binárneho predikátu „*p(s,o)*“, pričom :

[71] Resource Description Framework: http://www.w3.org/TR/rdf-primer/
[72] RDF Schema: http://www.w3.org/TR/rdf-schema/
[73] Web Ontology Language: http://www.w3.org/TR/owl2-primer/
[74] SPARQL Protocol and RDF Query Language: http://www.w3.org/TR/rdf-sparql-query/
[75] Rule Interchange Format: http://en.wikipedia.org/wiki/Rule_Interchange_Format
[76] Semantic Web Rule Language: http://en.wikipedia.org/wiki/SWRL
[77] Descriptive Logic: http://en.wikipedia.org/wiki/Description_logic
[78] Cryptography: http://en.wikipedia.org/wiki/Cryptography
[79] User Interface: http://en.wikipedia.org/wiki/User_interface

Obrázok 13: Graf jednej RDF trojice
Zdroj: http://www.w3.org/TR/rdf-primer/fig2dec16.png



Obrázok 14: Graf troch RDF trojíc
Zdroj: http://www.w3.org/TR/rdf-primer/fig3nov19.png



- daná trojica znamená : predikát *p* je vzťahom medzi *s(ubjektom)* a *o(bjektom)*
- každý z trojice entít je RDF meno, reprezentujúce nejaký prostriedok
- ak je entita URI, entita je braná ako konštanta, ak je entita literál, napr. „123“ môže byť
    - typová: *„123“^^xsd:integer*, kedy predstavuje výsledok mapovania hodnoty z pomedzi úvodzoviek do daného dátového typu, v tomto prípade teda číselnú hodnotu 123
    - netypová: „123“ kedy predstavuje daný doslovný text(aj s úvodzovkami), teda „123“

Na takúto trojicu, resp. tvrdenie je možné odkazovať sa explicitne. Súbor prepojených trojíc tvorí orientovaný graf, subjekty a objekty predstavujú uzly, predikát/vlastnosť[80] hrany. Na Obrázok 14 je príklad grafu takejto trojice

- Subjekt – stránka *„http://www.example.org/index.html“*
- má vlastnosť (predikát) – typu *„http://purl.org/dc/elements/1.1/creator“*
- s hodnotou (objektom) - ktorá je identifikovaná pomocou URI *„http://www.example.org/staffid/85740“*

Obrázok 13 je zasa graf troch takýchto trojíc

- Subjekt – stránka *„http://www.example.org/index.html“*
- má vlastnosti (predikáty)
    - Typu *„http://purl.org/dc/elements/1.1/creator“*
        - s hodnotou (objektom) - ktorá je identifikovaná pomocou URI *„ttp://www.example.org/staffid/85740“*

[80] Property

- Typu *„http://www.example.org/terms/creation-date“*
  - s hodnotou (objektom) - litrálom *„August 16, 1999“*
- Typu *„http://purl.org/dc/elements/1.1/language“*
  - s hodnotou (objektom) - litrálom *„en“*

Na ukladanie RDF dát, súbory trojíc/grafy, je možné použiť XML databáze, relačné databáze ale hlavne priamo nato určené grafové databáze, konkrétne ich podtyp *Triplestore*[81]. Serializovať RDF dáta, za účelom prenosu medzi napr. dvoja aplikáciami, je možné do viacerých syntaxí. Medzi najznámejšie patria asi RDF/XML[82], Turtle[83], Manchester[84].

4.2.RDF Schema Vocabulary Description Language

Napriek tomu, že RDF umožňuje reprezentáciu pomerne zložitých grafov, poskytuje iba malé prostriedky na sémantiku, automatické odvodzovanie, obmedzenia a atď.. RDFS rozširuje RDF o primitíva umožňujúce viac všeobecné znalosti. Medzi ne patrí napríklad trieda[85], podtrieda[86], prostriedok[87], literál, dátové typy (prebrané z *XML Schema*[88]), pod-vlastnosť[89], doména[90] a rozsah[91] pre vlastnosti, komentár[92], popis[93] atď. . Aj napriek pridaným novým prostriedkom je jazyk RDFS pre reprezentáciu znalostí na webe stále príliš slabý. Neumožňuje napr. definovať nutné a postačujúce podmienky pre prislúchanie do triedy, kardinality a negáciu.

O RDF vlastnostiach môžeme zamýšľať ako o atribútoch zdrojov, niečo ako tradičné dvojice *atribút-hodnota*. Taktiež prezentujú vzťahy medzi zdrojmi. RDF ale neposkytuje žiaden mechanizmus pre popis týchto vlastností a vzťahov. To je úlohou RDFS, ktoré definuje triedy a vlastnosti, ktoré môžu byť použité na popis tried, vlastnostní a zdrojov. RDFS neposkytuje slovník vlastností typu *„kontakt“*, poskytuje mechanizmus ako popísať skupiny vzájomne súvisiacich zdrojov a vzťahov medzi nimi. Tieto popisy sú písané v RDF.

---

[81] TripleStore: http://en.wikipedia.org/wiki/Triplestore
[82] RDF/XML: http://www.w3.org/TR/rdf-syntax-grammar/
[83] Turtle: http://www.w3.org/TeamSubmission/turtle/
[84] Manchester syntax: http://www.w3.org/TR/owl2-manchester-syntax/
[85] Class
[86] Subclass
[87] Resource
[88] XML Schema: http://en.wikipedia.org/wiki/XML_schema
[89] SubProperyOf
[90] Domain
[91] Range
[92] Comment
[93] Label

Dôležitým požiadavkom pre jazyk na prezentáciu znalostí na webe je štandardizované odvodzovanie, teda aby *odôvodňovače*[94] s rovnakým vstupom vrátili rovnaký výstup. Ale keďže odôvodňovanie musí byť efektívne a *rozhodnutľné*[95], musí byť takýto jazyk limitovaný vo výpočetnej zložitosti a vyjadrovacej sile. Jazyk musí taktiež nájsť správnu rovnováhu medzi otvorenou náturou a expanzívnosťou webu, oproti formálnej a dobre definovanej sémantike. Mal by stavať na existujúcich technológiách ako XML, RDF\S, umožňovať autorom rozširovať a znovapoužívať znalosť už existujúcu na webe, čo znamená ako možnosť importu ontologických súborov, ale aj branie v úvahu, že vytváraný súbor môže byť importovaný a rozširovaný. V konečnom dôsledku, každá ontológia si musela uvedomovať, že nie je kompletná a *odôvodňovače* mohli svoju prácu zakladať iba na výrokoch, ktoré boli pravdivé.

4.3.OWL Web Ontology Language

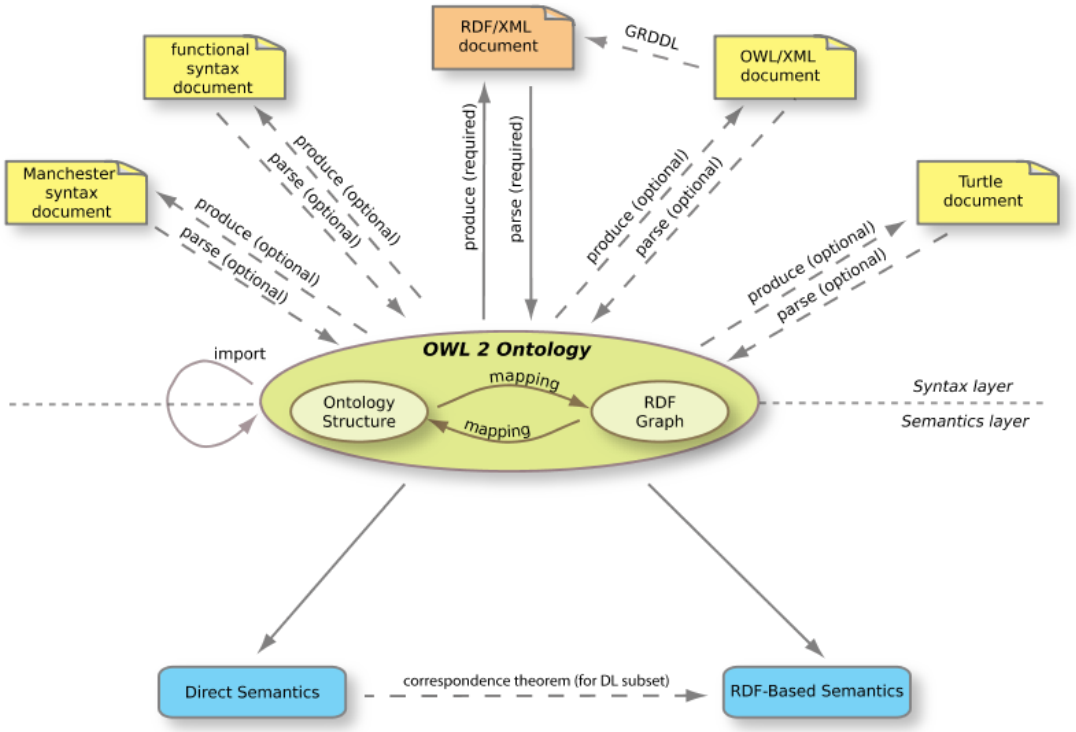
Prvým jazykom navrhnutým priamo na reprezentáciu znalostí na webe bol jazyk *Simple HTML Ontology Extension language*[96] (ďalej ako SHOE), umožňujúci sémantickú anotáciu stránok. Mal XML syntax a bol rámcovo založeným. Ďalším bol *DAML+OIL*[97], ktorý bol rozšírením RDFS a XML. Ako prvý kombinoval princíp zdieľania znalostí, formálnu sémantiku a limitujúcu expresívnosť DL, ktorá povoľovala úplnosť a spoľahlivosť. DAML+OIL je považovaný za prvý náležitý ZR jazyk určený pre web. Kombináciou rozšírenej sémantiky a syntaxe RDFS, DL sémantiky DAML+OIL a predpísania spôsobu publikovania a importu ontológií na webe zo SHOE, vznikol jazyk OWL.

Rôznorodé požiadavky pre jazyk pre ontológie na webe viedol k špecifikácií 3 verzií OWL: *Full*, *DL* a *Lite*. *OWL Full* verzia rozširuje sémantiku RDFS a ontológie v oboch jazykoch si sú navzájom konzistentné. *OWL DL*, syntaktická podmnožina Full verzie, ktorá je maximálne expresívnou rozhodnuteľnou DL, pre ktorú boli známe efektívne algoritmy. Z tohto dôvodu zachovávala iba kompatibilitu s podmnožinou RDFS. *OWL Lite* verzia, syntaktická podmnožina OWL DL, sa snažila obmedziť expresívnosť OWL DL, ktorá by umožňovala komplexné výrazy. V konečnom dôsledku sa ale zistilo, že aj napriek obmedzeniam je väčšina OWL DL sémantiky vyjadriteľná pomocou OWL Lite konštrukcií.

---

[94] Reasoner: http://en.wikipedia.org/wiki/Semantic_reasoner
[95] Decidable: http://en.wikipedia.org/wiki/Decidability_(logic)
[96] SHOE: http://en.wikipedia.org/wiki/Simple_HTML_Ontology_Extensions
[97] DAML+OIL: http://www.w3.org/TR/daml+oil-reference

Obrázok 15: Štruktúra OWL
Zdroj: http://www.w3.org/TR/owl2-overview/OWL2-structure2-800.png



4.3.1. OWL 2

Tento nasledovník OWL ho rozširuje o niekoľko vlastností, pre ktoré existujú efektívne odôvodňovacie algoritmy, a ktoré spĺňajú skutočné potreby užívateľov. Má modulárnu a rozšíriteľnú architektúru vo forme profilov, užívateľsky definovaných dátových typov a sémantickej anotácie. Rieši niekoľko nejednoznačností, rozširuje slovník o skrátené notácie často používaných kombinácií primitív v OWL, rozširuje expresívnosť vlastnostných axiómou, napr. o *asymetriu* (disjunktnosť vlastností – A > B vylučuje platnosť B > A), *reťazové zahrnutie vlastností* (napr. A vlastní B, zároveň vyjadruje, že A vlastní časti B)., *reflexívnost* (A je časťou samého seba) a *ireflexívnosť* (akcia A nemôže byť vlastnou príčinou), *kvalifikované obmedzenie kardinality* a iné, vylepšuje možnosti importu jednej ontológie do druhej a ich verzovanie.

Existujú 2 alternatívne spôsoby priradenia významu ontológiám v OWL 2 (víz. Obrázok 15): *OWL 2 DL* alias *Priama sémantika*[98] a *OWL 2 FULL* alias *Sémantika so základom RDF*[99]. OWL 2 DL je syntakticky obmedzená verzia OWL 2 FULL, ktorá zaručuje rozhodnuteľnosť a teda „*zjednodušuje život*“ napríklad implementátorom odôvodňovačov. OWL 2 FULL je zasa rozšírením RDFS a sleduje jeho filozofiu syntaxe, teda všetko je *trojica*. To umožňuje využitie napríklad už existujúcich technológií ako sú grafové databáze.

[98] Direct Semantic: http://www.w3.org/TR/owl2-direct-semantics/
[99] RDF-Based Semantic: http://www.w3.org/TR/owl2-rdf-based-semantics/

Skutočnosť, že väčšina existujúcich ontológií má tendenciu používať iba konkrétnu podmnožinu štruktúr jazyka a to, že proces odôvodňovania môže byť radikálne zrýchlený obmedzením expresívnosti jazyka, viedlo k vytvoreniu tzv. *profilov*, čo sú podmnožiny OWL 2 DL. Tieto podmnožiny sú špecifické tým, že sú obmedzené na syntaxi a poskytujú logiku, ktorá zvláda aspoň nejakú zaujímavú množinu služieb v polynomiálnom čase. Ide o

- OWL 2 EL – vyvinutú za účelom pokrytia niekoľkých už existujúcich veľkých ontológií, ako SNOMED CT, *Gene*[100] a *Galen*[101]. Hlavne pre potreby veľkého množstva klasifikácie a veľkých hierarchií.

- OWL 2 QL – prispôsobená pre možnosť prevodu axiómou do SQL dotazov, teda možnosť využitia existujúcich technológií relačných databáz.

- OWL 2 RL – profil založený na *Description Logic Programs*[102], ktoré umožňujú interakciu medzi DL a "pravidlami"[103]. Optimalizovaná na výkon odôvodňovačov.

4.4.SPARQL

Posledný z popisovaných prvkov SWS je jazyk pre dotazovanie sa nad RDF dátami, SPARQL. Základný tvar dotazu má tvar je popísaní v Tabuľka 3: Základný tvar SPARQL dotazu. SPARQL okrem veľa iného umožňuje :

- definovať prefixy pre skrátený zápis
  - `PREFIX kniha: http://www.kniha.com/onto`
- Zadať 4 typy dotazov
  - SELECT – výber surových dát
  - CONSTRUCT – vytvorenie nového RDF grafu
  - ASK – položiť dotaz s odpoveďou Áno/Nie
  - DESCRIRE – výber RDF grafu popisujúceho výsledok dotazu
- zadávanie podmienok
  - povinných – `{ ?y kniha#autor "Král" . }`
  - nepovinných – `{OPTIONAL { ?y kniha#autor "Král" . }}`
- robiť nad podmienkami

---

[100] Gene Ontology: http://www.geneontology.org/
[101] Galen: http://www.openclinical.org/prj_galen.html
[102] Description Logic Programs: http://www.cs.man.ac.uk/~horrocks/Publications/download/2003/p117-grosof.pdf
[103] Viac o pravidlách na http://ruleml.org/ alebo http://en.wikipedia.org/wiki/Rule_Interchange_Format

- konjunkciu
  - ```
    { ?y kniha#autor  "Král" .
      ?y kniha#vydavetel  "Mladé letá" . }
    ```
- zjednotenie
  - ```
    { { ?person foaf:mbox ?mbox }
    UNION
    { ?person foaf:mbox_sha1sum ?mbox } }
    ```

- robiť podmienky na hodnoty
  - ```
    { ?y kniha:rokVydania ?vydanie.
    FILTER xsd:date(?vydanie)>="25.5.2005"^^xsd:date }
    ```
- a mnoho ďalšieho[104]

Základný tvar dotazu v je zachytený v Tabuľka 3.

Časť dotazu	Čo sa v nej definuje	Príklad
PREFIX	Definovanie prefixov *menných priestorov*[105]	`PREFIX kniha: <http://www.kniha.com/onto>`
SELECT	Definovanie tvaru výsledky	`SELECT ?nazov ?autor`
FROM	Definovanie zdroja dát, nad ktorými sa dotazujeme	`FROM <http://www.kniha.com/kniznica.rdf>`
WHERE	Definovanie podmienok, ktoré musia vybrané dáta spĺňať	`WHERE {?nazov kniha:autor ?autor.}`
ORDER BY	Zoradenie výsledku podľa určitého parametru	`ORDER BY ?autor`

Tabuľka 3: Základný tvar SPARQL dotazu

4.5.RDFa

V nasledujúcich kapitolách sa stretneme niekoľko krát s pojmom *RDFa*[106], preto bude vhodné si ho vysvetliť ešte v rámci tejto technickejšej kapitoly. RDFa je skratka pre „*RDF in*

[104] SPARQL Query Language For RDF: http://www.w3.org/TR/rdf-sparql-query/
[105] Namespace: http://en.wikipedia.org/wiki/Namespace_%28computer_science%29#XML_namespace
    ```

attributes“, teda *RDF vo vlastnostiach*. Je to technológia, ktorá umožňuje vkladanie RDF dát priamo do HTML stránok bez toho, aby nejako ovplyvnili nejako ich doterajší vzhľad. Na získavanie RDF dát z *(X)HTML*[107] zdrojových kódov stránok sa používa *GRDDL*[108], čo je *odporúčanie W3C*. Momentálne existuje *XSLT*[109] implementácia, ale samotné GRDDL je dostatočne abstraktné a umožňuje aj iné implementácie.

[106] RDFa: http://www.w3.org/TR/xhtml-rdfa-primer/
[107] XHTML: http://en.wikipedia.org/wiki/XHTML
[108] GRDDL: http://www.w3.org/TR/grddl-primer/
[109] XSLT: http://en.wikipedia.org/wiki/XSLT

Tabuľka 4: Vrchná hierarchia SNOMED CT

• Klinický nález (*Clinical finding*) • Procedúry (*Procedure*) • Pozorovateľná entita (*Observable entity*) • Štruktúra tela (*Body structure*) • Organizmus (*Organism*) • Látka (*Substance*) • Farmaceutický/biologický produkt (*Pharmaceutical / biologic product*) • Vzorka (*Specimen*) • Špeciálny koncept (*Special concept*) • Väzbový koncept (*Linkage concept*) • Fyzická sila (*Physical force*)	• Udalosť (*Event*) • Prostredie alebo geografická poloha (*Environment or geographical location*) • Sociálny kontext (*Social context*) • Situácia s explicitným kontextom (*Situation with explicit context*) • Štádium a rozsah (*Staging and scales*) • Fyzický objekt (*Physical object*) • Kvalifikátor hodnoty (*Qualifier value*) • Záznamový artefakt (*Record artifact*)

5. SNOMED CT

Ako už bolo spomenuté v úvode, vo svete dochádza každoročne k mnoho zbytočným úmrtiam a úrazom spôsobeným zlou komunikáciou medzi zdravotníkmi, nedôkladnými/nejednoznačnými lekárskymi správami, zábudlivosťou zaneprázdnených alebo únavou pracovníkov. Takýmto udalostiam je možné predísť. Dodanie štandardu pre klinickú terminológiu pre použitie naprieč svetom zdravotných informačných systémov by mohlo značne prispieť k zvýšeniu kvality a bezpečnosti v zdravotníctve. Prehľadnejšie a jednotným spôsobom zapísané zdravotné záznamy zlepšia komunikácia a inter-operabilitu v zdravotných systémoch a umožnia vznik systémom pre podporu rozhodovania v zdravotníctve.

Definícií toho, čo SNOMED CT[110] (ďalej ako SCT) je existuje veľa. Vymenujem z nich aspoň pár, z môjho pohľadu, najdôležitejších a najvýstižnejších. SCT je

- komplexná klinická terminológia, ktorá poskytuje klinický obsah a vyjadrovanie pre klinickú dokumentáciu a reportovanie. Môže byť použitá na kódovanie, získavanie a analýzu klinických dát.
    - Niektorými ľuďmi je dokonca považovaná za najkomplexnejšiu multi-jazyčnú klinickú terminológiu sveta.
- je systematicky organizovaná a počítačovo spracovateľná kolekcia medicínskej terminológie, ktorá pokrýva väčšinu oblastí medicíny, ako sú choroby, nálezy, výkony, mikroorganizmy, látky atď. Umožňuje konzistentné indexovanie, ukladanie,

[110] Systematized Nomenclature of Medicine - Clinical Terms

získavanie a agregovanie klinických dát a hlavne pomáha organizovať medicínske záznamy, redukuje rôznorodosť ich zápisu. Umožňuje konzistentnú výmenu informácií a je základom pre inter-operatibilné elektronické zdravotné záznamy

- ontológia klinický pojmov a výrazov.

SCT je obsahovo rozdelený do hierarchií (víz. Tabuľka 4).

5.1.Logická štruktúra

Terminológia sa skladá z konceptov, popisov a vzťahov.

5.1.1. Koncepty

V kontexte SCT, je koncept klinický výraz, identifikovaný unikátnym numerickým identifikátorom – *ConceptId* (ďalej ako CID). Je prezentovaný jedinečným ľudsky čitateľným *plne špecifikovaným menom* (*Fully Specified Name)* (ďalej ako FSN). Koncepty sú formálne definované ich vzťahmi (*relationships*) s ostatnými konceptmi. Táto logická definícia dáva konceptom explicitný význam, ktorý môžu počítače spracovať a dotazovať sa naň. Každý koncept má taktiež množinu výrazov, ktoré ho pomenúvajú v ľudský čitateľom tvare. Môžu predstavovať všeobecnejšie pojmi, napr. *„procedúra"* (*procedure*), alebo špecifickejšie, napr. *„excizionálna biopsia lymfatických uzlín"* (*excisional biopsy of lymph node*).

5.1.2. Popisy

Každému konceptu je priradených niekoľko (2-n) popisov (*descriptions*), ktoré sú v ľudsky čitateľnom tvare tvorené výrazmi alebo názvami. Každý popis má vlastný jedinečný identifikátor – *DescriptionID* (ďalej ako DID). Každý popis je priradený práve 1 konceptu. Napríklad popisy pre koncept s CID *„22298006"* sú

- *Fully Specified Name: |Myocardial infarction (disorder)| DescriptionId 751689013*
- *Preferred term: |Myocardial infarction |DescriptionId 37436014*
- *Synonym: |Cardiac infarction |DescriptionId 37442013*
- *Synonym: |Heart attack |DescriptionId 37443015 ...*

Existuje niekoľko typov popisov. Ako už bolo spomenuté, každý koncept má práve 1 unikátne *plne špecifikované meno* (*Fully Specified Name*). Malo by popisovať koncept a vyjasniť jeho význam. Nie je používané pri ľudskom vyjadrovaní. Každé je ukončené výrazom v zátvorke, sémantickou značkou, ktorá napomáha zjedinečniť koncept medzi konceptmi, ktoré môžu používať rovnaký výraz. Napríklad:

- *„Hematóm (morfologická abnormalita)“* (*Hematoma (morphologic abnormality))* - reprezentujúca *„hematóm, ktorý vidí patológ na úrovni tkaniva“ (the 'hematoma' that a pathologist sees at the tissue level*)
- *„Hematóm (porucha)“* (*Hematoma (disorder)*) - reprezentuje *„klinická diagnóza, ktorú spraví lekár ak sa rozhodne, že osoba má 'hematóm'“* (*the clinical diagnosis that a clinician makes when they decide that a person has a ‚hematoma'*)

Každému konceptu v rámci daného dialektu jazyka je priradený jeden *preferovaný výraz* (*Preferred term*) (ďalej ako PF), nemusí byť jedinečný. Je to výraz bežne používaný pre daný koncept.

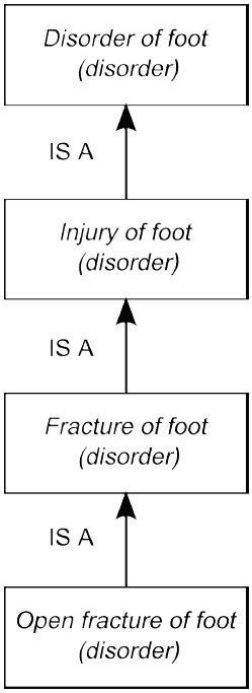
Synonymum je ďalší typ popisu, ktorý môže byť použitý pre reprezentáciu daného konceptu. Je nepovinný, ale jeden koncept ich môže mať priradených viac.

5.1.3. Vzťahy

Spájajú koncepty a modelujú tvrdenia o nich. Vzťahy sa vytvárajú iba ak tvrdenie, ktoré reprezentujú je stále pravdivé. Každý vzťah je reprezentovaní 3 konceptmi : zdroj, cieľ a typ vzťahu. Rozdeľujú sa do 4 základných typov :

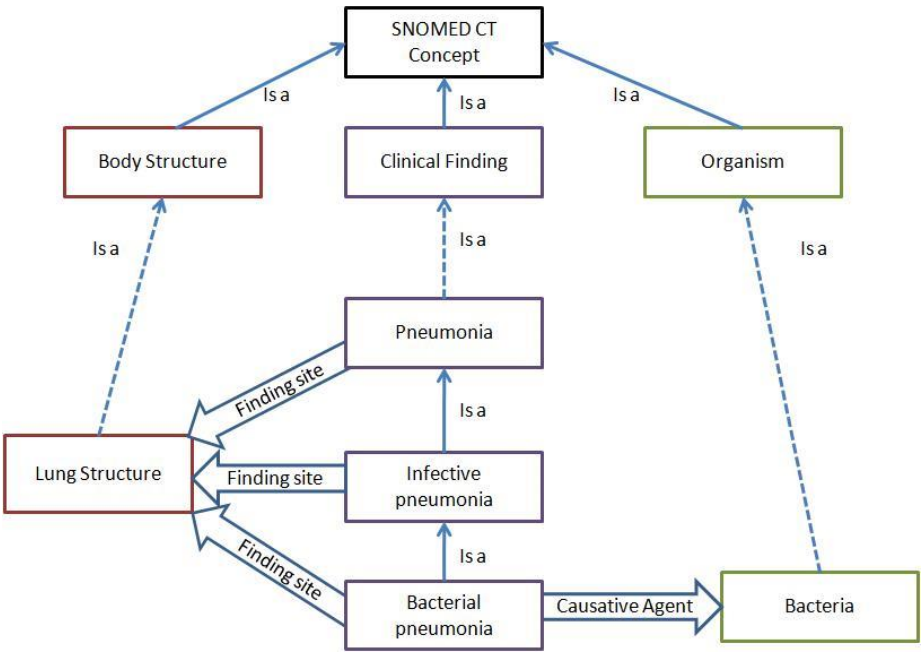
- **kvalifikujúci** (*qualifying*)
- **historický** (*historical*)
- **doplnkový** (*additional*)
- **definujúce** (*defining*)

Najdôležitejším z týchto typov sú definujúce vzťahy (víz. príklad Obrázok 17). Každý koncept je logicky definovaný pomocou jeho definujúcich vzťahov s ostatnými konceptmi. Každý aktívny koncept, okrem hlavného konceptu, má minimálne jeden definujúci *„is a“* (víz. Obrázok 16) vzťah so svojim(i) nadtypom(pmi) a môže mať niekoľko definujúcich atribútových vzťahov, ktoré modelujú rozdiely oproti jeho nadtypom.



Obrázok 16: Príklad niekoľkých na seba naviazujúcich *„is a*" vzťahov tvoriacich hierarchiu
Zdroj: (14 s. 23)

Obrázok 17: Príklad definujúcich vzťahov
Zdroj: (14 s. 25)



5.1.3.1. Atribútové vzťahy

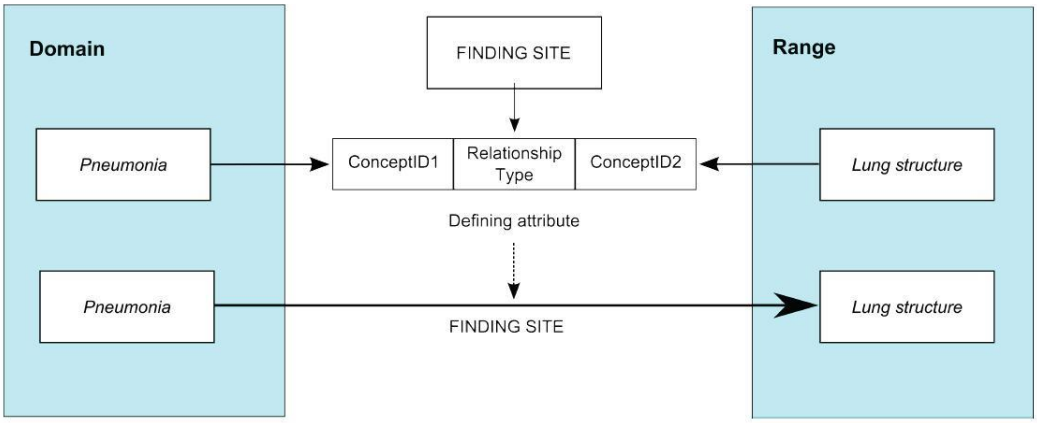
Každý atribútový vzťah má *zdroj* (*source*), *meno (typ vzťahu)* a *hodnotu (cieľ - destination)* vzťahu. SCT obsahuje 50 typov atribútových vzťahov. Väčšinu z nich je možné aplikovať iba v rámci jednej hierarchie konceptov. Hierarchie, na ktoré je možné daný atribútový vzťah aplikovať, sa nazýva *doména* (množina zdrojov) atribútu. Napríklad atribút *„Prepojená morfológia“* (*ASSOCIATED MORPHOLOGY*) môže byť použití na hierarchii *„Klinické nálezy“* (*Clinical finding*) ale nemôže byť použitý na hierarchii *„Procedúra“* (*Procedure*). Atribút má zároveň určenú obmedzenú množinu hodnôt (cieľových konceptov), ktorú nazývame *rozsah* (*range*). Na Obrázok 18 je tento vzťah graficky zobrazení. Napríklad pre spomínaný atribút *„Prepojená morfológia“* je rozsah *„Morfologicky abnormálna štruktúra“* (*Morphologically abnormal structure (morphologic abnormality)* |) a jeho podtypy. Samotné atribúty medzi sebou vytvárajú hierarchiu. Popisovať konkrétnu štruktúru tejto hierarchie atribútov/konceptov, uvádzať konkrétne zoznamy atribútov/konceptov by pre tento text nebolo prínosom a nie je ani jeho cieľom, preto v prípade záujmu čitateľa odporúčam preštudovať si túto problematiku napríklad v užívateľskej príručke SCT (14).

5.2.Fyzická štruktúra

Samotný SCT je distribuovaný v textových súboroch, v ktorých sú textové výrazy oddelené znakom tabulátora. Fyzická štruktúra SCT pozostáva z

- SCT dátové komponenty a ich vzťahy, vrátane *jadrových* (*core*) štruktúr

Obrázok 18: Príklad domény a rozsahu
Zdroj: (14 s. 28)



- História (*History*)
- Podmnožiny (*Subsets*)
- Krížové mapovanie (*Cross Mapping*)
- Rozšírenia (*Extensions*)

Vzťahy medzi týmito stavebnými blokmi znázorňuje Obrázok 19.

5.2.1. „*Core*" tabuľky

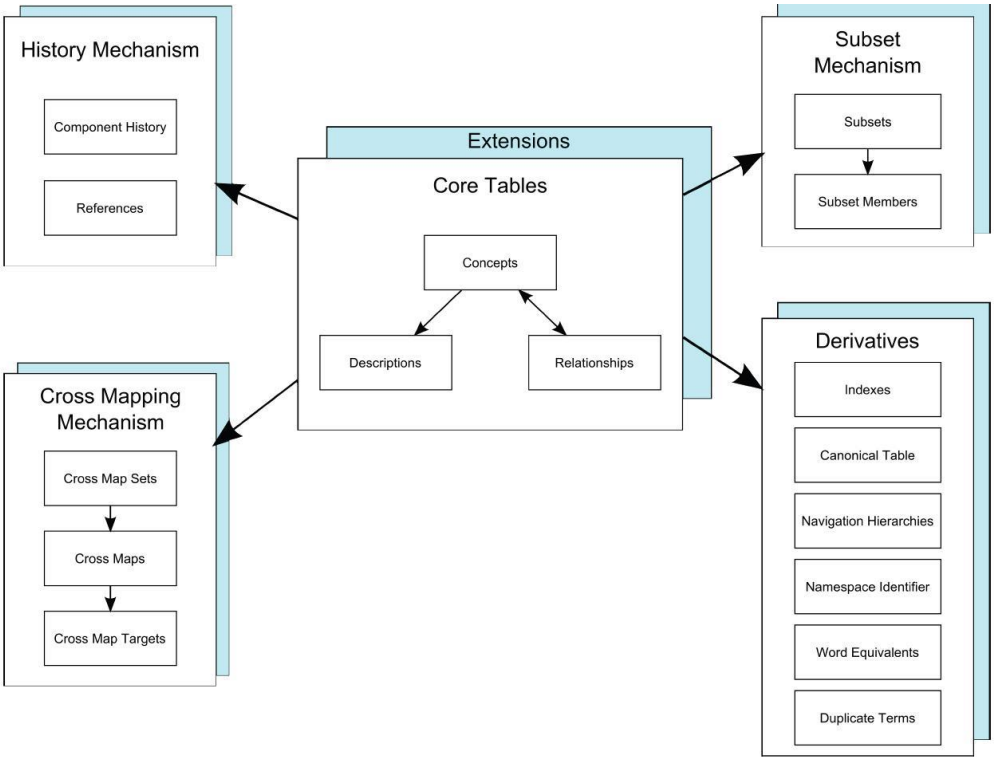
Jadrové tvoria tabuľky konceptov, popisov a vzťahov. Vzťahy medzi nimi sú zachytené na Obrázok 20. Asociácie medzi tabuľkami sú tvorené CID, ktorý je používaný ako *primárny*[111] ale aj ako *cudzí kľúč*[112]. V tabuľke konceptov existuje pre každý koncept práve jeden riadok, ktorý obsahuje :

- *CID* - jedinečný a nemenný identifikátor konceptu
- *FSN* - plne špecifikujúce meno
- *Status konceptu* (*ConceptStatus*) - indikujúci aktívnosť alebo zastaranosť konceptu. Zastarané koncepty sú uchovávané kvôli spätnej kompatibilite
- *SnomedID* (ďalej ako SID)ToDo_F - ďalší unikátny identifikátor v rámci všetkých komponent v rámci celého SCT
- *Indikátor zložitosti* (*IsPrimitive*) - určuje „*jednoduchosť*" konceptu, vlastnosť využívaná v aplikáciách, ktoré pracujú s deskriptívnou logikou

Tabuľka popisov, výrazy použite pre pomenovanie konceptov, obsahuje pre každý popis práve jeden riadok s údajmi :

[111] Primary Key: http://en.wikipedia.org/wiki/Unique_key
[112] Foreign Key: http://en.wikipedia.org/wiki/Unique_key

Obrázok 19: Vzťahy medzi základnými stavebnými blokmi SCT
Zdroj: (14 s. 81)

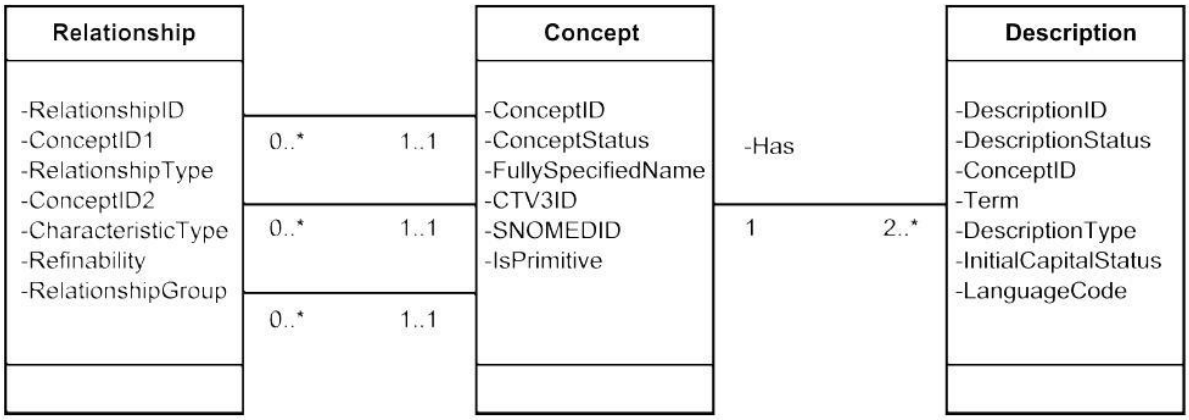


- *DID* - jedinečný a nemenný identifikátor popisu.
- *Typ popisu* (*DescriptionType*) - indikujúci, či daný popis predstavuje FSN, preferovaný názov alebo synonymum.
- *Kód jazyka* (*LanguageCode*) - určuje jazyk, alebo dialekt jazyka(napr. *UK English*), ktorým je popis písaný.
- *Status popisu* (*DescriptionStatus*) - indikátor aktívnosti, resp. dôvodu zostarnutia
- *Výraz* - samotný výraz
- *Veľkosť úvodného písmena výrazu* (*InitialCapitalStatus*) - či má byť pri použití výrazu prvé písmeno veľké

Každý vzťah medzi konceptmi je uvedený práve v jednom riadku tabuľky vzťahov. Vzťah, ako už bolo uvedené, je reprezentovaný 3 komponentmi : zdrojový koncept, koncept určujúci typ vzťahu, cieľový koncept(aká hodnota). Každý riadok v tabuľke obsahuje tieto informácie :

- *Identifikátor* (*RelationshipId*) (ďalej ako RID) - jedinečný a nemenný identifikátor vzťahu
- *CID1* - identifikátor zdrojového konceptu. Teda napríklad potomok „IS A“ vzťahu, alebo zdroj atribútového vzťahu
- *Typ vzťahu* (*CID*) (*RelationshipType*) - identifikátor konceptu určujúci typ vzťahu.

Obrázok 20: Vzťahy medzi tabuľkami konceptov, popisov a vzťahov
Zdroj: (14 s. 79)



- *CD2* - identifikátor cieľového konceptu. Teda napríklad rodič „*IS A*" vzťahu, alebo hodnota atribútového vzťahu
- *Charakteristický typ* (*CharacteristicType*) - indikátor, či vzťah predstavuje definujúcu alebo kvalifikujúcu charakteristiku zdrojového konceptu
- *Refinability* - indikátor toho, či je možné spresniť cieľový koncept ak je daný vzťah použitý ako šablóna ...
- *Vzťahová Skupina* (*RelationshipGroup*) - číselná hodnota, ktorá zoskupuje vzťahy do logicky asociovaných vzťahových skupín

5.2.2. Historické tabuľky

SCT prechádza neustálim vývojom. Vznikajú nové komponenty, komponenty zastarávajú alebo sú aktualizované. Komponenty, ktorých použitie už nie je žiadané (zmena významu, vývoj, oprava chyby, atď.) sú označené za neaktívne a ich opätovné použitie bude znemožnené. Tieto zmeny sú zaznamenávané do historických tabuliek:

- *Historická tabuľka komponent* (*Component history table*) (ďalej ako CHT) - uchováva zmeny v statusoch
    - SID / *Verzia „releasu"*(*ReleaseVersion*) / *Typ zmeny* (*ChangeType*) / *Status* / *Dôvod* (*Reason*)
- *Referenčná tabuľka komponent* (*Component references table*) (ďalej ako CRT) uchováva referencie z neaktívnych komponent na ich aktuálne aktívne nahradzujúce alebo súvisiace komponenty. Každý záznam uchovávaj aj podstatu medzi danými komponentmi
    - *CID* / *Typ referencie* (*ReferenceType*) / *Identifikátor referencie* (*ReferenceId*) (ďalej ako RID)

- CID môže ukazovať na koncept iba v prípade že daný koncept je stave *„presunutý inam“* (*MovedElseWhere*). Zmeny v konceptoch sú majoritne zaznamenávané v tabuľke vzťahov
- RID zasa ukazuje na koncept iba v prípade zneaktívnenia popisu

5.2.3. Pravidlá a typy zmien

Zmeny v SCT sa riadia sadou pravidiel.

- Pridanie novej komponenty – je zachytené v záznamom v CHT o pridaní a novým záznamom v odpovedajúcej DT.
- Zmena statusu komponenty – pridanie záznamu v CHT s CID, typ zmeny ako *„Zmena statusu“* (*StatusChange*) a nový status komponenty. V prípade zmeny konceptu alebo popisu nastane úprava stavu danej komponenty v odpovedajúcej DT.
- Zmena vzťahu – by mohla ovplyvniť význam uložených dát. Žiaľ, SCT tieto zmeny nijako nezaznamenáva. Jediná možnosť vysledovania zmien vo vzťahoch je porovnávania RT z rôznych *„releasov“*. To isté platí aj o entitách v podmnožinách, krížových mapách.
- *„Nevýznamné zmeny“* - je skupina úprav od ktorých nie je požadované zneaktívnennie a pridávanie nových komponent. Ide o zmeny, ktoré nemenia význam komponenty, napr. zmena kapitalizácie názvu, úprava výslovnosti, atď.. Ich uskutočnenie napriek tomu požaduje záznam v CHT
- *„Významné zmeny“* – vyžadujú zneaktívnenie komponent a pridanie nových, nahradzujúcich. Podľa toho, o ktorú komponentu sa jedná, sú potrebné ešte dodatočné záznamy :
  - Popis / Podmnožina /Krížna mapa – pridanie záznamu do CHRT
  - Koncept – záznam v RT zachytávajúci vzťah medzi zneaktívneným a nahradzujúcim aktívnym konceptom

Opätovné zaktívnenie komponenty je možné. U konceptov nie je nutné, aby boli znovu zaktívnené všetky pôvodné definujúce vzťahy. Popisy daného konceptu musia byť tiež zaktívené.

5.2.4. Podmnožiny

Podmnožiny sú množiny komponent, ktoré sú príslušné k určitému jazyku, špecializácií, organizácií atď., čo predstavuje typ danej podmnožiny. V podstate ide zoznam CID. Ich účelom je vytváranie celkov pozostávajúcich iba z určitých častí SCT, ktoré využíva určitá

skupina ľudí. Niektoré sú dokonca súčasťou oficiálnej distribúcie SCT. Podmnožiny používajú predpísanú štruktúru 2 tabuliek :

- *Tabuľka podmnožín* (*Subset table*) (ďalej ako ST) - každý riadok predstavuje „*release*“ podmnožiny. Obsahuje zoznam podmnožín, ktoré sú obsiahnuté v *členskej tabuľke podmnožín* (*Subset member table*).
    - *Identifikátor podmnožiny* (*SubsetID*) (ďalej ako SID) / *Pôvodný identifikátor podmnožiny* (*OriginalId*) (ďalek ako OID) – jedinečný identifikátor „*releasu*“ podmnožiny/podmnožiny
    - *Verzia* / *Meno* / *Typ podmnožiny* (*SubsetVersion/Name/Type*), *Kód jazyka* (*LanguageCode*), *Identifikátor oblasti* (*RealmId*), Identifikátor kontextu (*ContextId*)
- *Členská tabuľka podmnožín* (*Subset member table*) (ďalej ako SMT) - každý riadok predstavuje jedného člena podmnožiny, koncept alebo popis. Obsahuje 4 hodnoty :
    - *SID* / *Identifikátor člena* (*MemberId)* (jedno z *CID / DID / RID*)
    - *Status člena* (*MemberStatus*) / LinkedId - účely týchto prvkov sa líšia podľa typu podmnožiny

Podmnožiny nevytvárajú vlastné komponenty, iba sa odkazujú na existujúce, v *jadrových* tabuľkách.

5.2.5. „Cross mapping“

Vo voľnom preklade „*krížové mapovanie*“ umožňuje SCT odkazovať na inú terminológiu, klasifikáciu atď. Pomocou krížových máp je možné koncept mapovať do *cieľového schématu* ako jedno z :

- jeden kód cieľového schématu
- kolekcia kódov, ktoré dokopy reprezentujú daný koncept
- manuálna voľba z množiny možností pre mapovanie do odpovedajúceho alebo podobného výrazu

Krížové mapovanie ale neumožňuje mapovať množinu konceptov do množiny, resp. do jediného kódu v cieľovom schémate. Pre podporu krížového mapovania obsahuje SCT 3 tabuľky :

- *Tabuľka krížových máp* (*Cross Map Sets Table*) (ďalej ako CMST) - kde každý riadok reprezentuje cieľové schéma, pre ktoré existuje krížové mapovanie

- *Tabuľka krížovej mapy* (*Cross Maps Table*) (ďalej ako CMT) - každý riadok obsahuje jednu možnosť pre mapovanie konceptu do cieľového schématu
- *Tabuľka cieľov krížovej mapy* (*Cross Map Targets Table*) (ďalej ako CMTT) - každý riadok reprezentuje kód, alebo množinu kódov v cieľovom schémate

5.2.6. Zvyšné súčasti

Okrem už popísaných častí obsahuje SCT ešte :

- *Rozšírenia* - Napriek tomu, že SCT je pomerne komplexná terminológia, niektoré organizácie alebo skupiny ľudí potrebujú pre svoje potreby dodatočné vlastnosti. Mechanizmus rozšírení umožňuje týmto skupinám rozšíriť SCT o komponenty, referenčné množiny a podmnožiny (teda majú vlastné komponenty, vlastné podmnožiny, teda aj vlastné fyzické súbory, teda aj vlastné *jadrové* tabuľky).
- *Sada nástrojov pre vývoj* (*Developement toolkit*) - Pomôcky pre zjednodušenie práce vývojárov. Napr. indexy pre vyhľadávanie, hierarchie pre zobrazovanie alebo traverzovanie.
- *Kanonická tabuľka* - používaná pre zisťovanie ekvivalencie konceptov, ktoré môžu byť reprezentované viacerými spôsobmi.

5.2.7. SNOMED CT Identifier(SCTID)

Každá komponenta SCT je identifikovaná a referencovaná pomocou identifikátoru, ktorý je dátového typu *SNOMED CT Indetifier* (SCTID). Je to 64 bitový celočíselný typ, ktorému pravidlá pre jeho generovanie zaručuje, že je pre každú komponentu jednoznačný a jedinečný. Umožňuje vytvárať intervaly / partície identifikátorov pre špecifické použitie, napr. pre isté typy komponent alebo menné priestory pre organizácie atď.. Pre SCTID platí :

- má minimálne 6 cifier a max 18
- je bez-znamienkový, teda iba kladné čísla
- nenesie sémantickú informáciu, ale môže niesť štrukturálnu
- posledná cifra je kontrolná, slúži na kontrolu správnosti celého ID
- 2. cifra od konca určuje typ entity, koncept, popis, vzťah, podmnožina, množina / cieľ z krížovej mapy
- 3. cifra od konca identifikuje, či ide o entitu zo hlavnej časti SCT alebo z jeho rozšírenia
- v prípade, že entita patrí do hlavnej časti SCT, je zvyšných 3 až 15 cifier určených pre identifikátor entity
- v prípade, že entita patrí do rozšírenia SCT

- cifry na mieste 4 až 11 (7cifier) z prava identifikujú menný priestor organizácie, ktorá je povolená vytvárať rozšírenia
- cifry na miestach 12 až 18 je určených pre identifikátor entity v rámci rozšírenia danej organizácie

5.3.Prevod SNOMED CT do OWL

„*Stated*“ tvar konceptu, je logická definícia konceptu, s ktorou pracujú autori a editori. Pozostáva z minimálne 1 „*stated*“ „*IS A*“ vzťahu a 0..N definujúcich vzťahov, ktoré existujú ešte pred behom klasifikátora nad logickými definíciami (ktorý pravdepodobne pridáva dodatočné odvodené vzťahy). *„stated" vzťahová tabuľka* (*Stated relationship table*) (ďalej ako SRT) s rovnakým formátom ako RT, obsahuje iba tieto „*stated*" vzťahy, pričom identifikátor vzťahu nie je uvedený. Identifikátor vzťahu tvorí trojica konceptov, definujúca samotný vzťah. Distribúcia SCT obsahuje *Perl*[113] zdrojový kód, ktorý SRT prevádza na OWL.

Stredný koncept vzťahovej trojice, atribút, resp. všetci potomkovia konceptu *„410662002 |concept model attribute|(CMA)*“, odpovedajú OWL *objektovým vlastnostiam* (*Object properties*). *„IS A*“ vzťahy medzi konceptmi v podstrome CMA v SRT sú skriptom prevádzané na vytváranie axiómu *pod-vlastnosti* (*sub-property*).

Každý vzťah, ktorý má povolené zoskupovanie (atribút vzťahu), musí byť vnorený pod existenčné obmedzenie, ktoré predstavuje tú skupinu, čo je dosiahnuté vlastnosťou objektu „*RoleGroup*“.

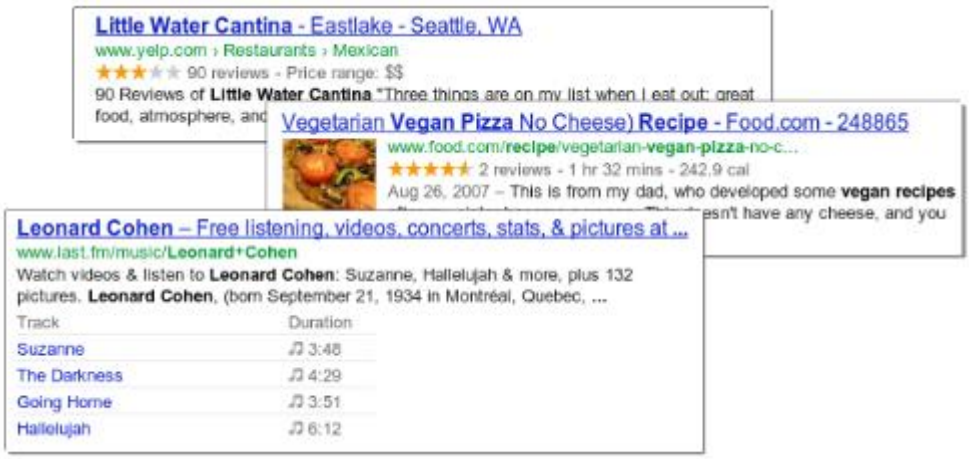
[113] Perl: http://en.wikipedia.org/wiki/Perl

6. Názor : Sémantický web a ontológie

Aj keď je táto technológia s nami už pomerne dlho, doposiaľ mi nepríde ako dostatočne vyspelá nato aby ju svet (hlavne teda internetový) plnohodnotne prijal. Veľmi dobrým krokom k tomuto je prijatie jazyka OWL konzorciom W3 ako štandard pre tvorbu ontológií. Žiaľ aj túto skutočnosť zatieňuje niekoľko negatív. Možnosť, respektíve nutnosť, výberu správnej sémantiky medzi OWL DL a OWL FULL alebo jedného z profilov OWL EL / OWL QL / OWL RL mi príde v tomto smere pre ľudí kontraproduktívna, pretože väčšina užívateľov nemá záujem skúmať túto problematiku do hĺbky aby sa neskôr v nejakom výbere mohla rozhodnúť správne, aby náhodou zlým výberom neznehodnotili svoju neskoršiu prácu. Ľudia si chcú iba nainštalovať program a začať ho používať bez zbytočných a hlavne takto dôležitých výberov. Je pravda, že od samotnej syntaxe by mal byť užívateľ odtienený nástrojom, ale vývojári dobre vedia, že automaticky generovaný kód je potreba sem tam skontrolovať či nástroj nevygeneroval náhodou zbytočný kód, resp. vôbec kód, ktorý užívateľ chcel. V prípade dát uložených v databáze je možne skontrolovať záznamy v DB, v prípade práce so serializovanými dátami je možné nahliadnuť na kód generovaný nástrojom. Momentálne mám rozhľaď o existencii minimálne 5 syntaxi. Učiť sa všetkých 5, alebo tú s ktorou pracuje práve používaný nástroj, nie je ideálny stav.

Ako už bolo spomenuté v úvode, neexistencia alebo iba malé množstvo dostatočne jednoduchých a užívateľsky prívetivých nástrojov pôsobí tiež ako „*brzda*“ rozšírenia tejto technológie.

Obrázok 21: *Rich Snippet* vo výsledkoch vyhľadávania na Google
Zdroj: http://www.google.com/help/hc/images/webmasters/webmasters_99170_rsreview_en.png



V oblasti tvorby webových stránok nie je situácia až taká zlá. Na webe je k dostupnosti veľa nástrojov, alebo *„pluginov“* (*plug-in*[114]) do overených nástrojov, ktoré umožňujú tvorbu sémantizovaných stránok. Dokonca aj najznámejšie *CMS*[115] systémy ako je *WordPress*[116], *Drupal*[117] alebo *Joomla*[118] už majú alebo už pracujú na podpore *microformátov*[119], *microdata*[120] alebo *RDFa*[121] atď.. Zaujímavým nástrojom je napríklad *online* editor RDFa stránok, *RDFaPlay*[122], ktorý pracuje *real-time* a umožnoje zobrazenie samotnej vytváranej stránky, serializovaný zápis RDF dát na vytváranej stránke a hlavne vizualizovaný RDF graf. Zaujímavý je aj validátor RDF dát s možnosťou vizualizácie, priamo od *W3C*[123]. Pre vývoji ontológií existuje tiež niekoľko vyspelých nástrojov, kde medzi najvyspelejšie a najkomplexnejšie, keď nie práve najvyspelejší, patrí *Protégé*[124]. Ponúka možnosti importu ontológií, vizualizácie, odôvodňovanie, možnosť pokladať SPARQL dotazy atď.. Nová verzia 4.0 pracuje už kompletne s OWL. Za spomenutie stoja napríklad nástroje *NeOn Toolkit*[125] alebo *TopBraid Composer*[126]. Medzi zaujímavé *„online“* editory by som zaradil *WebProtege*[127], *BioPortal*[128], *Knoodl*[129], kde posledné 2 menované ponúkajú oproti prvému aj vizualizáciu grafu vytváranej ontológie.

Ako bolo zmienené v texte vyššie, prijatie OWL ako de-facto štandardu je vec, ktorá túto technológiu posúva o niečo dopredu. Neexistujú ale ontológie, ktoré by boli *„odbornou verejnosťou“* prijaté za štandard napríklad len v doméne danej ontológie. Takto je užívateľ znova nútený k odbornejšiemu štúdiu a hľadaniu správnej (alebo nedajbože vytvoreniu vlastnej) ontológie, čo je pomerne odradzujúce. Neexistuje teda žiadna *„super-ontológia“* pokrývajúca všetku znalosť ľudstva. Ani populárne a často používané ontológie ako *FOAF*[130]

---

[114] Plug-in: http://en.wikipedia.org/wiki/Plug-in_(computing)
[115] Content Management System: http://en.wikipedia.org/wiki/Content_management_system
[116] WordPress: http://wordpress.org/
[117] Drupal: http://drupal.org/
[118] Joomla: http://www.joomla.org/
[119] Microformat: http://en.wikipedia.org/wiki/Microformat
[120] Microdata: http://en.wikipedia.org/wiki/Microdata_(HTML)
[121] RDFa (in attributes): http://en.wikipedia.org/wiki/RDFa
[122] RDFa Play: http://rdfa.info/play/
[123] W3C Validation service: http://www.w3.org/RDF/Validator/
[124] Protégé: http://protege.stanford.edu/
[125] Neon Toolkit: http://neon-toolkit.org/wiki/Main_Page
[126] Top Braid Composer: http://www.topquadrant.com/composer/
[127] WebProtege: http://webprotege.stanford.edu/
[128] BioPortal: http://bioportal.bioontology.org/
[129] Knoodl: http://www.knoodl.com/ui/groups/knoodl/wiki/Help/entry/Examples
[130] FOAF: http://www.foaf-project.org/

(zachytenie sociálnych vzťahov) alebo *Dublin Core*[131] (popis digitálnych objektov ako hudba, obrázky, knihy) nemožno nazvať za štandard vo svojej doméne.

Existuje ale aj veľa pozitív, kvôli ktorým by sa web vývojári mali o sémantický web začať zaujímať čim skôr a čím viac. Medzi takéto dôvody patria napríklad

- *Rich snippets*[132] (Google) – zobrazovanie útržkov relevantných informácií už vo výsledkoch vyhľadávania. Google tieto informácie identifikuje vďaka sémantickým značkám v zdrojových kódoch stránok. Založený hlavne na microdata formáte, ale podporuje aj RDFa a microformat. Viz **Error! Reference source not found.**
- *Open Graph Protocol*[133] (Facebook) (ďalej ako OGP) – ak chcete aby Vaša stránka získavala „*lajky*“, ak chcete aby sa Váš článok bol odporúčaný iným užívateľom, ak chcete aby bola Vaša špecialita v reštaurácií označená a propagovaná za chutnú na Facebooku, v podstate sa chcete dostať do *Facebok Social Grafu* (víz. odstavec nižšie), čo je dosiahnuteľné niekoľkými sémantickými značkami. Založený na microformátoch a RDFa.
- *SCHEMA.ORG*[134] (spoločná iniciatíva Bing, Google, Yahoo!, Yandex) – Microdata slovník vyvíjaný v spoločným úsilím spomenutých firiem za účelom zlepšenia vyhľadávania a identifikácie obsahu stránky. Podporuje primárne microdata, ale postupne aj RDFa.

Ako je vidieť všetky tieto technológie podporujú priamo, alebo pridávajú spätne podporu, RDFa. Preto je môjho názoru vhodnejšie držať sa pri prípadnej implementácií sémanticky na svojich stránkach práve RDFa, ktoré umožňuje jednoduchým spôsobom elementu stránky priradiť viacero vlastností z viacerých slovníkov (ontológií) a tým si zabezpečiť podporu u každej zo spomenutých iniciatív.

Už niekoľko krát spomínaný veľký *internetový hráči* ako Google, Microsoft alebo Facebook nezaliehali a sémantické technológie využívajú (interne) aktívne už teraz. Medzi zaujímavé a už fungujúce projekty patria napríklad

---

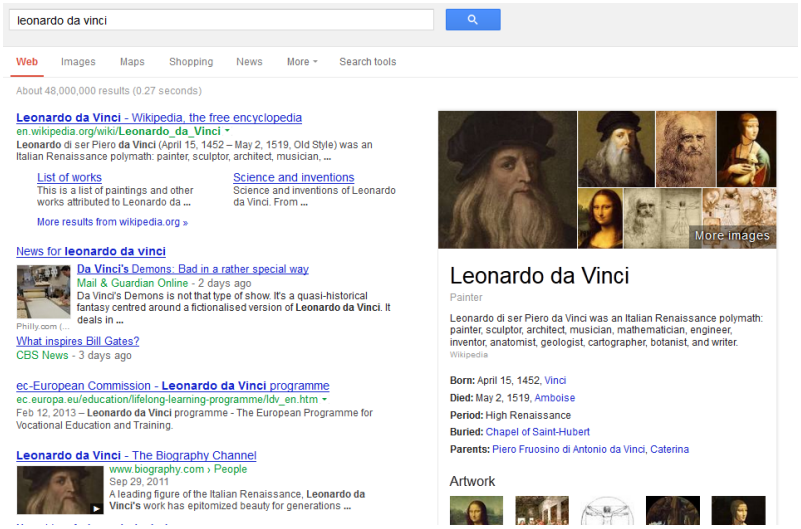
[131] Dublin Core: http://dublincore.org/

[132] Rich Snippets: http://googlewebmastercentral.blogspot.cz/2009/05/introducing-rich-snippets.html

[133] Open Graph Protocol: http://ogp.me/

[134] Schema.org: http://schema.org/

Obrázok 22: Google knowledge graph
*Zdroj: **http://www.google.cz/search?q=leonardo+da+vinci***



- *Knowledge graph*[135] (Google) – je graf, ktorý vo svojom pozadí buduje Google a ktorý už teraz obsahuje cez 500 miliónov objektov (konceptov), cez 3,5 miliárd faktov (axiomou) / vzťahov medzi týmito objektmi, kde každý objekt predstavuje nejakú skutočne existujúci vec našom reálnom svete, ako je napríklad *Eiffelová veža*[136]. Tento graf je iniciatíva Google o prechod z hľadania zhodou textu k vyhľadávaniu informácií o skutočných objektoch reálneho sveta. Google s týmto plánuje vylepšiť svoje vyhľadávanie tým, že zobrazí výsledky týkajúce sa iba danej entity, alebo daného významu daného viac-významového slová (tzv. homonyma), zobrazenie základných a najčastejšie vyhľadávaných informácií o danej entite hneď vo výsledku, atď.. Google v podstate na pozadí vytvára obrovskú ontológiu *„o tom, čo vie“*. Výsledok je viditeľný už teraz v pravej časti stránsky s výsledkom hľadania (viz Obrázok 22: Google knowledge graph).
- *Social graph*[137] (Facebook) – na pozadí celého Facebooku je funguje tzv. *Sociálny graf*, ktorý v sebe nesie informácie o spojeniach medzi entitami ako sú osoby (teda ich profily), fotkami, geografickými miestami atď.. Sprístupnením OGP umožnil Facebook užívateľom určitým spôsobom tento graf ovplyvňovať. Samotný graf funguje na sémantických technológiách.
- *Foaf.sk* [138] – zaujímavý projekt postavený na sémantických technológiách a s myšlienkou, ktorá mi značne pripomína projekt *„Open data initiative*[139]“, teda

[135] Knowledge Graph: http://www.google.com/insidesearch/features/search/knowledge.html alebo http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html
[136] Eiffel Tower: http://en.wikipedia.org/wiki/Eiffel_Tower
[137] Social Graph: http://en.wikipedia.org/wiki/Social_graph
[138] http://foaf.sk/

sprístupnenie dát rôznych inštitúcií pre verejnosť a pre rôzne voľné použitie. Využíva voľne prístupné dáta slovenských štátnych inštitúcii ako je obchodný register alebo daňový úrad a zobrazuje informácie o podnikateľských subjektoch, ich pôsobení vo firmách, napojenie na iné subjekty, existujúce dlhy atď.

[139] Open Data Initiative: http://www.opendatainitiative.org/ alebo pekné video : http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.html

7. Manažment systém pre ontológie

Praktická časť tejto práce by sa mala zamerať na návrh a implementáciu manažment systému pre ontológie (ďalej ako OMS). Jadro aplikácie by malo byť postavené okolo **editoru a prehliadača ontológií**. Medzi vlastnosti takéhoto systému, pre ktoré existuje už množstvo praxou overených riešení / návrhových vzorov / postupov, ale ktoré sú zároveň čo sa týka aktuálnej témy nezaujímavé a riešia nutnú funkcionalitu multi-užívateľského systému, patrí napríklad správa užívateľov, ich právomocí, bezpečnosť systému, rozšíriteľnosť, plug-iny atď. Pri implementácií aplikácie sa preto primárne zameriam na vlastnosti, ktoré by vedeli obdobný software spraviť ešte viac príťažliví pre jeho užívateľov po stránke práce s ontológiami a spomenutej nutnej funkcionalite pre viac-užívateľský systém sa budem venovať druhorado.

7.1.Funkcionalita OMS

7.1.1. ODP

Funkcionalita týkajúca sa ODP (viz. Návrhové vzory) by bola pre užívateľov veľmi užitočná. **Knižnica znova-použiteľných ODP**, do/z ktorej by bolo možné ODP pridávať/odoberať, by mohla prácu na vývoji ontológií rapídne zrýchliť. Samotná knižnica by bola zbytočná, ak by nebolo v editore možné ODP ľahko používať, teda **vkladať do editovanej ontológie** a napr. **z označených uzlov editovanej ontológie ODP vytvoriť a uložiť do knižnice**. Administrácia vzorov v knižnici je samozrejmosť. Ak si ontológiu predstavíme ako graf, tak ODP, čo je v podstate malá ontológia, je malý graf a jeho vloženie by mohlo byť realizované ako súčasné vloženie množiny uzlov a hrán (celého pod-grafu), s možnosťou namapovať žiaden/niektoré/všetky uzly z vkladanej množiny do už existujúcich uzlov grafu editovanej ontológie.

7.1.2. Vytváranie ontológií

V predchádzajúcich kapitolách boli spomenuté viaceré spôsoby vytvárania novej ontológie. Asi základným typom je práca s jednou ontológiou, teda **začatie prací na *„zelenej lúke”***, alebo na existujúcej ontológií, teda uloženej po poslednej editácií alebo na jednej zo sady základných kostrových ontológií, ktoré sú vhodné ako *základné stavebné kamene* ontológii. Práca s viacero ontológiami by zahŕňala **import (časti) ontológie** do editovanej ontológie alebo vytvorenie novej ontológie **integráciou viacerých vhodných existujúcich (podmnožín) ontológií**. Pri práci s množným číslom ontológií by bolo vhodné využiť funkcionalitu, ktorá už bude musieť existovať pre prípad funkcionality okolo práce s ODP.

Nie len pri vývoji rozsiahlych ontológií je vhodné ich **verzovanie**. Verzovanie ontológií je ešte pomerne otvorená téma, na ktoré ešte neexistuje nejaké štandardné alebo zaužívané riešenie.

7.1.3. Využitie odôvodňovača

Každý editor by mal poskytovať **služby** nejakého **odôvodňovača**. OMS by nemal byť výnimkou a mal by využívať nejaký odôvodňovač minimálne pre 2 účely. Prvou je odvodzovanie vzťahov, ktoré sú vyvodené z manuálne vytvorených. Tie by malo byť možné zároveň zobraziť, čo by bola voliteľná funkcia. Kontrola integrity ontológie by mala byť druhou. Ako príklad tejto funkcionality je zistenie či niektoré triedy nie sú nerealizovateľné. Oba tieto funkcionality by mali uľahčiť prácu tvorcovi ontológie. Skôr menovaná tým, že v grafe zobrazí vzťahy, ktoré autorovi nemusia byť „voľným okom“ viditeľné a môžu byť nechcené, čo sa pri zväčšovaní ontológie stane veľmi rýchlo. Neskôr zmienená zasa umožní nájsť (nie len) triedy, ktorých vzťahy spôsobujú ich nerealizovateľnosť.

7.1.4. Viacjazyčnosť, užívatelia, protokoly

Systém by mal byť v niekoľkých smeroch viacjazyčný. Prvý z nich je samotné prostredie, ktoré by malo užívateľovi ponúkať prepnúť komplet prostredie do vybraného dostupného jazyka. Toto rozhodnutie by si systém pre jednotlivých užívateľov mal samozrejme pamätať. Samotné pridanie nového jazyka prostredia by pre administrátora malo predstavovať jednoduchý úkon.

Práca na ontológií nemusí predstavovať iba pridávanie a uberanie vzťahov / tried / indivíduí, ale aj napríklad pridávanie prekladov v rôznych jazykoch pre už existujúce entity v nejakom základnom jazyku ontológie. S týmto typom činnosti môže súvisieť viac funkcionalít. Za účelom **viacjazyčnosti** samotných editovaných **ontológií** bude stačiť použitie iba tzv. *jazykových značiek* pre OWL označenia[140], *SKOS*[141] by mi iba za týmto účelom prišiel ako zbytočný „kanón na vrabce“. Na úkon samotného pridania prekladu by mohlo byť naviazané viacero akcií. Jednou z nich by mohli byť odporúčania na ďalší, resp. dodatočný preklad množín súvisiacich pojmov. Identifikácia týchto množín na preklad by mohla byť riešená algoritmicky napríklad na základe počtu vzťahov medzi entitami, alebo na základe pozorovaní správania užívateľov

[140] OWL Annotations: http://www.w3.org/TR/owl-ref/#Annotations
[141] SKOS: http://www.w3.org/TR/skos-reference/

- pri „browsovaní“ ontológiou – entity, ktorými sa k prekladanej entite užívatelia často dostávali, alebo entity, ktoré boli navštívené prechodom od prekladanej
- činnosti prekladu – entity prekladané často spoločne s prekladanou entitou v iných jazykoch

Nie len pridanie prekladu, ale rôzne typy úprav ontológie ako pridanie, odobratie alebo zmena entity, by mohli pred „oficiálnym“ zavedením do ontológie podliehať schvaľovaciemu procesu. Jeden nemenný protokol samozrejme nemôže vyhovovať každej skupine vývojárov alebo typom zmien. Výber z niekoľkých protokolov, alebo možnosť editácie či dokonca vytvárania vlastných by predstavoval veľké plus, ale žiaľ samotný editor schvaľovacích protokolov a ich prípadné automatické implementovanie je netriviálna úloha sama o sebe.

Pre-rekvizitou spomenutého protokolu, ale v podstate aj každého viac-užívateľského prostredia, je autorizačný systém pre užívateľov. V tomto smere považujem za vhodné použiť zaužívaný a praxou overený spôsob, tzv. *„Access control list“*[142] (ďalej ako ACL), ktorého základný princíp je založený na tabuľke obsahujúcej informácie o prístupových právach. Typicky záznam takejto tabuľky obsahuje subjekt (užívateľ / užívateľská skupina / rola), typ operácie a objekt, čo v preklade znemená, že daný subjekt ma právo previesť daný typ operácie s daným objektom. Základným príkladom pre tento zoznam môže byť rola editora ontológie a schvaľovateľa zmien nad danou ontológiou. Ak by sme sa vrátili k téme viacjazyčnosti, ďalším príkladom pre nutnosť zvolenia správnej granuality, by mohli byť role *„prekladateľov*“ a *„schvaľovačov / garantov*“ prekladov entít do jazykov. Jednotlivý prekladatelia a garanti by mali mať právo prekladať / schvaľovať iba určité typy prekladov a to podľa východzieho a cieľového jazyka prekladu, resp. podľa expertnej oblasti do ktorej entity patria. Potrebám systému by malo byť podriadené zvolenie správnej veľkosti granuality typov operácií a objektov, ktoré bude možné v tejto tabuľke zachytiť. V prípade vhodne zvolenej granuality by sme neskôr spomenutú dvojicu rol mohli považovať z jemnejšie nastavené skôr spomenutej, teda prekladateľ ako editor, ktorý pridáva entitám iba jazykové značky, schvaľovateľ prekladov zasa ako schvaľovateľa editácie jazykových značiek. Už pri popise schvaľovacích protokolov a ACL boli spomenuté výrazy ako užívateľ, užívateľská rola a skupina užívateľov, teda potreba systému rol (skupina oprávnení) a skupín užívateľov je nevyhnutná.

---

[142] Access Control List : http://en.wikipedia.org/wiki/Access_control_list

7.2.OMS a SNOMED CT

Pôvodný úmysel bol OMS do určitej miery prispôsobiť ontológií SNOMED CT. Súčasťou distribúcie SNOMED CT je aj Perl skript, ktorý prevádza iba časť tejto ontológie do OWL. Veľká časť komponent SNOMED CT je teda špecifická iba pre túto ontológiu. Žiaľ natívny tvar SNOMED CT je značne špecifický a prispôsobenie sa jeho tvaru a špecifickým komponentom by znamenalo úplne znemožnenie práce s inými ontológiami. Možnosť práce s OWL a teda s viacerými ontológiami bol hlavný dôvod prečo som sa rozhodol skôr pre podporu štandardného OWL ako špecifického natívneho tvaru SNOMED CT.

7.3.Editor a prehliadač „*all in one*“

Hlavné prívlastky, ktoré by mali editor definovať, sú grafický a intuitívny. Jeho podstata by mala byť v graficky spracovanom editore násobného orientovaného grafu. Vďaka tomu bude funkciu prehliadača spĺňať samotný editor so znemožnenou možnosťou editácie. Základnými funkciami, ktorými bude editor a vizualizér disponovať, sú:

- **Načítanie/ukladanie** – ontológií z/do
    - interného úložištia systému(pravdepodobne grafová DB)
    - súboru na lokálnom/internetovom úložišti

  a v podporovaných syntaxiach
- **ODP** - už vyššie spomenutá funkcionalita vkladania a ukladania ODP
- **Módy zobrazovania ...**
    - **vzťahov** – možnosť zapnutia/vypnutia zobrazovania vzťahov napr.:
        - podľa typu vzťahu (napr. iba hierarchických)
        - získaných odvodením odôvodňovačom
    - **celkového vzhľadu / pohľadu** – niektoré formy zobrazenie sú jednoducho lepšie pre účel
        - prezeranie veľkých ontológií
        - náhľadu na detail nejakej ontológie
        - rôzne rozloženia uzlov (hviezdicovité, hierarchické, stromové...)
        - atď...
    - možnosť zoskupiť určitú časť pod-grafu do jedného uzlu a jeho opätovné rozloženie do pôvodnej skupiny samostatných uzlov
    - možnosť zachytenia pohľadu na ontológiu – teda nie len samotné uloženie ontológie, ale aj samotného rozloženia uzlov v aktuálnom okne editora nejakého užívateľa s možnosťou tento pohľad „preposlať“ inému užívateľovi
- Detailné informácie – zobrazovanie detailných informácií o vzťahoch, vlastnostiach, bodoch...
    - v nápovedách tzv. „tool-tip“
    - v oknách nato určených (s možnosťou editácie)
    - Zobrazenie zoznamu miest použitia danej entity v rámci ontológie

- Kontextové menu – možnosť práce na ontológií pomocou informačných okien, ale aj pomocou kontextového menu a iných grafických prvkov jednotlivých entít v grafe. Napr.:
    - Vytváranie vzťahov spájaním uzlov (tried) v grafe
    - vytvárania nových uzlov (tried) v grafe z kontextového menu
    - Mazanie entít priamo z grafu
    - Práca s anotáciami
    - atď.
- Možnosť nastavenia vzhľadu a rozloženia okien editoru

Tabuľka 5: Technológia X Prostriedky

	Skúse-nosti	Vývojové nástroje	Web FW	Grafová DB (Triplestore)	Odôvodňo vač	užitočný SW
JAVA	Malé	*NetBeans*[143], *Eclipse*[144] (*STS*[145]), IntelliJ[146]	*Spring*[147], *GWT*[148], *JSF*[149]	*Jena*[150], *Sesame*[151], *Neo4j*[152], *Joseki*[153]	*Fact++*[154], *Pellet*[155], *HermiT*[156], *TrOWL*[157]	*OWL API*[158], *owlpopulous*[159] *Jena, Sesame*
PHP	Dobré	Eclipse, NetBeans	*Symfony*[160], *Nette*[161]	*ARC2*[162]	--	*Redland*[163], Graphite[164]
.NET	Veľmi dobré	*MVS*[165] (*Express*[166])	*ASP .NET MVC*[167]	*BrightstarDB*[168], *SemWeb-DotNet*[169], *dotSesame*[170]	--	*dotNetRDF*[171], *linqtordf*[172], *Simple OWL.Api*[173]

[143] NetBeans: http://netbeans.org/
[144] Eclipse: http://www.eclipse.org/
[145] Spring Tool Suite: http://www.springsource.com/developer/sts
[146] IntelliJ: http://www.jetbrains.com/idea/
[147] Spring Framework: http://www.springsource.org/
[148] Google Web Kit: https://developers.google.com/web-toolkit/
[149] JavaServer Faces: http://javaserverfaces.java.net/
[150] Apache Jena: http://jena.apache.org/
[151] Sasame: http://www.openrdf.org/
[152] Neo4j: http://neo4j.org/
[153] Joseki: http://joseki.sourceforge.net/
[154] Fact++: http://owl.man.ac.uk/factplusplus/
[155] Pellet: http://clarkparsia.com/pellet/
[156] HermiT: http://hermit-reasoner.com/
[157] TrOwl: http://trowl.eu/
[158] OWL API: http://owlapi.sourceforge.net/
[159] Owlpopulous: http://code.google.com/p/owlpopulous/
[160] Symfony Framework: http://symfony.com/
[161] Nette Framework: http://nette.org/
[162] ARC2: https://github.com/semsol/arc2/wiki
[163] Redland: http://librdf.org/
[164] Graphite: http://graphite.ecs.soton.ac.uk/
[165] Microsoft Visual Studio: http://www.microsoft.com/visualstudio/11/en-us
[166] Microsoft Visual Studio Express: http://www.microsoft.com/visualstudio/11/en-us/products/express
[167] ASP .NET MVC: http://www.asp.net/mvc
[168] BrightstarDB: http://www.brightstardb.com/
[169] SemWeb.NET: http://razor.occams.info/code/semweb/
[170] dotSesame: http://sourceforge.net/projects/dotsesame/
[171] dotNetRdf: http://www.dotnetrdf.org/
[172] LinqToRdf: http://code.google.com/p/linqtordf/

7.4.Návrhové rozhodnutia

7.4.1. Desktop verzus Web aplikácia

Základnou, nie len architektonickou, otázkou pred samotným návrhom muselo byť rozhodnutie či bude editor OMS *desktopovou*[174] alebo *webovou aplikáciou*[175]. Oba typy aplikácií majú svoje výhody a nevýhody medzi ktorými sa iba ťažko vyberá. Tento software by mal priniesť niečo nové a keďže desktopové aplikácie pre tvorbu ontológií, ktoré dosahujú skutočne vysokú úroveň už existujú (napr. Protégé v4.1, TopBraid Composer), rozhodol som sa pre aplikáciu spustiteľnú v internetovom prehliadači (ďalej ako browser), ktorých zastúpenie je vcelku malé a s ktorými práca nie je až taká príjemná ako s tými desktopovými.

7.4.2. Platforma

Za výberom technologickej platformy boli hlavné kritéria skúsenosti s vývojom pod danou platformou, dostupnosť a vyspelosť vývojových prostredí pre danú platformu a existencia programov nutných k vytvoreniu spomínaného OMS, ako sú frameworky, existencia vhodnej databáze (resp. databázového konektoru) a odôvodňovača. *„Mainstreamovými"* platformami pre vývoj webových aplikácií sú v súčasnosti *Java*[176], *PHP*[177], *.NET*[178], *Ruby*[179]. Tabuľka 5 ukazuje kombináciu týchto platforiem s ich kritérií výberu. Aj napriek väčším skúsenostiam s vývojom aplikácií pod platformou *.NET* a *PHP*, musím výber platformy podriadiť kritériu dostupnosti a existencie softwarových komponent ako je odôvodňovač, databáze a sémantické frameworky a ich vlastností ako je veľkosť komunity za danou komponentou, aktívny vývoj, podpora zo strany tvorcu a existencia skutočných projektov s použitím danej komponenty. V tomto zmysle je jasnou voľbou platforma *JAVA*.

7.4.3. Vizualizácia

Vizualizáciu v browseroch je možné riešiť niekoľkými spôsobmi. V nasledujúcom texte budem často používať výraz grafovo vizualizačný framework (ďalej ako GVF), ktorý by som prirovnal k nástrojom uľahčujúcim vývoj aplikácií pre vizualizáciu grafov.

---

[173] Simple OWL.Api: http://simpleowlapi.codeplex.com/
[174] Desktop Application: http://www.pcmag.com/encyclopedia_term/0,1237,t=desktop+application&i=41158,00.asp
[175] Web Application: http://en.wikipedia.org/wiki/Web_application
[176] JAVA: http://en.wikipedia.org/wiki/Java_(programming_language)
[177] PHP: http://en.wikipedia.org/wiki/PHP
[178] .NET Framework: http://en.wikipedia.org/wiki/.NET_Framework
[179] Ruby: http://en.wikipedia.org/wiki/Ruby_(programming_language)

- *Flash*[180] (*Flex*[181]) – platforma od firmy *Adobe*[182] pre vývoj multimediálnych a interaktívnych internetových aplikácií. Pre účel takéhoto editoru by bol vhodnou voľbou, žiaľ s touto technológiou nemám skúsenosti. Napriek tomu by som rád upozornil na GVF, ktorý pri prieskume veľmi zaujal *Flare*[183] a ktorý používa práve túto platformu.
- *Java applet*[184]– je aplikácia vytvorená na platforme Java, ktorá je pomocou Java Pluginu spustiteľná v browseroch ako *tenký klient*[185]. Existuje už niekoľko vyspelých GVF pre tento typ, napr. *Graphviz*[186], ktorý je používaný aj samotným editorom Protégé. Žiaľ kvôli závislosť na technológii, s ktorou nemám veľké skúsenosti som túto možnosť nebral v úvahu.
- *Microsoft Silverlight*[187] - je obdoba *Flash* ale od firmy *Microsoft*[188], postavená na platforme *.Net Framework*[189]. Jeho veľkou nevýhodou sú problémy s podporou na browseroch na OS[190] Linux[191], čo bolo asi jedným z hlavných dôvodov pre jeho nepoužitie.
- *JavaScript*[192] *& HTML(5)* – všetky doterajšie možnosti boli závislé na nutnosti inštalovania nejakého „*pluginu*" alebo celej sady nástrojov do browsera, resp. do na samotné PC. Táto varianta nepotrebuje okrem samotného browsera (dostatočne aktuálneho) žiadne dodatočné zdroje. Ide o kombináciu browserom štandardne podporovaného programovacieho jazyka *JavaScript* a štandardných prvkov jazyka HTML5 určených pre vizualizáciu, *Canvas*[193] a SVG[194]. Tento fakt bol jeden hlavných dôvodov, prečo prijať som za prostriedky pre vizualizáciu zvolil práve toto riešenie.

---

[180] Adobe Flash: http://en.wikipedia.org/wiki/Adobe_Flash
[181] Adobe Flex: http://en.wikipedia.org/wiki/Adobe_Flex
[182] Adobe: http://www.adobe.com/
[183] Flare: http://flare.prefuse.org/
[184] Java Applet: http://en.wikipedia.org/wiki/Java_applet
[185] Thin Client: http://en.wikipedia.org/wiki/Thin_client
[186] Graphviz: http://www.graphviz.org/
[187] Silverlight: http://en.wikipedia.org/wiki/Microsoft_Silverlight
[188] Microsoft: http://en.wikipedia.org/wiki/Microsoft
[189] .NET Framework: http://en.wikipedia.org/wiki/.NET_Framework
[190] Operating System: http://en.wikipedia.org/wiki/Operating_system
[191] Linux: http://en.wikipedia.org/wiki/Linux
[192] JavaScript: http://en.wikipedia.org/wiki/JavaScript
[193] HTML Canvas element: http://en.wikipedia.org/wiki/Canvas_element
[194] HTML SVG element: http://en.wikipedia.org/wiki/Scalable_Vector_Graphics

7.4.3.1. Canvas versus SVG versus Framework

Zvolené riešenie kombinácie Javascript a HTML5 so sebou prináša ešte potrebu dodatočného výberu HTML5 elementu, do ktorého chceme prípadný graf kresliť. Oba prvky sú svojimi vlastnosťami diametrálne odlišné :

- **SVG** – skratka pre *Scalable Vector Graphic*, jedná sa teda o vektorovú grafiku a všetko čo je na SVG element nakreslené sa stáva súčasťou *DOM*[195] a je možné *„povesiť" obsluhu udalostí*[196].
- **Canvas** – je prvok, na ktorý sa kreslia *pixely*[197]. Okamžite po nakreslení sa prvky stávajú iba skupinou pixelov, na ktorú nejde povesiť žiadnu obsluhu udalostí. Vďaka tomu je kreslenie na neho rýchle.

Koho by rozdiely medzi týmito dvoma prvkami zaujímali viac odporúčam napríklad krátky článok na stránke *Sitepoint*[198] a veľmi pekné rýchlostné porovnanie vzhľadom k počtu prvkov na elemente na stránkach *eleqtriq.com*[199]. Rozhodovanie sa medzi týmito dvoma elementmi by bolo zložité. Na jednej strane možnosť práce s DOM objektmi, na druhej rýchlosť v prípade veľkých zobrazovaných grafov, ktoré pri ontológiách hrozia. Z druhej strany, aký význam má zobrazený veľký graf s veľkým množstvom strán a uzlov? Koľko uzlov a strán naraz má človek možnosť pobrať v jednom momente? Rozuzlenie nakoniec nepriniesli porovnania v rýchlostiach a vlastnostiach týchto elementov, ale niečo úplne iné, a to existencia a schopnosti dostupných a existujúcich *javascriptových* GVF.

Internet je plný rôznych GVF. Stačí do Google zadať pojem *„javascript graph library"* alebo *„javascript graph framework"*. Výsledky sú plné rôznych odkazov na GVF, diskusné fóra kde podobný problém s hľadaním alebo výberom vhodného kandidáta riešilo už milión ľudí predo mnou, rôzne články s rebríčkami alebo zoznamom užitočných a odporúčaných GVF. Časom sa ale mená GVF opakujú dookola. Veľa z týchto GVF som vylúčil hneď na začiatku práve kvôli chýbajúcej sémantike vo vyhľadávaní. Anglické slovo *„graph"* (slovensky graf) má primárne význam [200]ako *usporiadaná dvojica (V, E), kde V je konečná množina vrcholov a* $E \subseteq \binom{V}{2} = \{\{x, y\} : x, y \in V, x \neq y\}$. Druhý, na internete v oblasti

[195] DOM: http://en.wikipedia.org/wiki/Document_Object_Model
[196] Event (handler): http://en.wikipedia.org/wiki/Event_(computing)#Event_handler
[197] Pixel: http://en.wikipedia.org/wiki/Pixel
[198] sitepoint: http://www.sitepoint.com/canvas-vs-svg-how-to-choose/
[199] Eleqtriq.com: http://www.eleqtriq.com/2010/02/canvas-svg-flash/
[200] Graph: http://en.wikipedia.org/wiki/Graph_(mathematics)

zobrazovania asi častejšie používaný, je význam *diagramu*[201] (anglicky *diagram* alebo *chart*), pre jednoduchosť si postačíme s definíciu grafickej reprezentácií dát. Vylúčené GVF boli teda primárne určené na zobrazenie druhého významu slova *„graph“*. Výsledný zoznam okrem iných obsahoval približne tieto GVF: (*g*[202])*Raphael*[203], *InfoVis*[204], *Processing.js*[205], *D3.JS*[206], *ProtoVis*[207], *Paper.js*[208], *jsViz*[209], *Google Chart Tools*[210], atď.. Po pokusoch so zobrazeniami, študovaní možností daných GVF, som vybral knižnicus *D3.JS*, hlavne kvôli jej programátorských možnostiam, dobrej dokumentácií, aktívnemu vývoju, veľkého množstva dostupných príkladov s grafovými štruktúrami.

7.4.4. Uloženie dát

Samotná aplikácia bude potrebovať ukladať niekoľko druhov dát, samotné dáta ontológií, metadáta portálu/editoru. Za metadáta portálu je možné považovať napríklad rôzne typy ACL, informácie o užívateľoch, osobné nastavenia užívateľov portálu, metadáta ontológií atď. Za týmto účelom bude vhodné použiť zaužívané spôsoby ich ukladania v relačných DB so schémami tomu uspôsobenými, v properties súboroch atď. Na samotnom výbere konkrétnej relačnej DB už toľko nezáleží, pretože vlastnosti dostupných relačných DB za účelom ukladania a prístupu k takému malému množstvu dát nie sú nijakým spôsobom pre užívateľa alebo vývojára postrehnuteľných reps. zásadným. U dát samotných ontológií už výber takýto *jednoduchý* a jednoznačný nie je.

7.4.4.1. Relačná vs Grafová databáza

Ontológie je možné ukladať viacerými spôsobmi. Za základné rozdelenie týchto spôsobov uloženia považujem to, či ide o uloženie serializovanej podoby ontológie v jednej z dostupných syntaxí, alebo či ide o uloženie v nejakom type databáze.

V prvom prípade, ako už bolo spomenuté v kapitole 4.1 RDF Resource Description Framework, máme k dispozícií niekoľko dostupných syntaxí, napríklad :

- OWL2 Functional Syntax

---

[201] Diagram: http://en.wikipedia.org/wiki/Chart
[202] gRaphael: http://g.raphaeljs.com/
[203] Raphael: http://raphaeljs.com/
[204] InfoVis: http://thejit.org/
[205] ProcessingJS: http://processingjs.org/
[206] D3JS: http://d3js.org/
[207] Protoviz: http://mbostock.github.com/protovis/
[208] Paper.js: http://paperjs.org/
[209] jsViz: http://www.jsviz.org/blog/
[210] Google Chart Tools: https://developers.google.com/chart/

- OWL2 XML Syntax
- Manchester Syntax
- RDF/XML Syntax
- RDF/Turtle

Toto uloženie mám využitie hlavne vo výmene dát medzi aplikáciami alebo v jedno užívateľských aplikáciách. Pre interaktívnu aplikáciu akou by OMS malo byť je vhodnejší druhý spomenutý spôsob uloženia.

Hlavnou výhodou uloženia v databázach je zrýchlenie práce s ontológiou. Niektoré databáze vznikli „na zelenej lúke“ priami za účelom ukladania tripletov, niektoré zasa iba ako nadstavba nad už existujúcimi (komerčnými) hlavne relačnými databázami. Aj napriek nižším nákladom na vývoj druhého typu DB, je prvá skupina z dlhodobého hľadiska kvôli rýchlosti, efektívnosti a implementovateľnosti efektívneho dotazovania nad grafovo založeným RDF modelom dát perspektívnejšia. Kvôli týmto dôvodom som sa rozhodol použiť v OMS jeden z dostupných triplestore databáz. Keďže pre manipuláciu s OWL dátami som zvolil framework Jena, využijem k uloženiu OWL dát triplestore dodávaný priamo v Jena frameworku, TDB. Framework Jena umožňuje ukladane OWL dát do rôznych DB, nebude v budúcnu problém zameniť TDB z nejakú inú grafovú alebo dokonca relačnú DB, ktorá bude v Jena frameworku podporovaná.

7.4.4.2. Práca s dátami

Pre prácu s dátami ontológií, odôvodňovačom, vyhodnocovaním SPARQL dotazov, som sa rozhodol použiť framework Jena[211] za účelom aspoň čiastočnej štandardizácie. Jedná sa komplexný semantický webový framework nie len pre vytváranie, manipuláciu a serializáciu OWL ontológií.

7.5.Návrh a implementácia aplikácie

Pred samotnou implementáciou prototypu aplikácie vznikol software design dokument (ďalej ako SDD). Dokument obsahuje návrh, architektúru, popis komponent a ich zapojenie, dátový model a jeho vysvetlenie, prípady užitia prototypu aplikácie atď. Keďže samotný SDD má rozmer väčší ako táto práca, bol vložený iba ako príloha na priloženom CD (víz. 13 Dodatok A: CD).

[211] Jena: https://jena.apache.org/

Implementácia aplikácia bola uvedená do stavu prvej verzie s prvými funkciami. Aplikácia, ako aj návod na jej provizórne spojazdnenie a krátka užívateľská príručka je uložená na priloženom CD (víz 13 Dodatok A: CD)

8. Záver

V práci sme si spočiatku predstavili pojem ontológie, ako jeho historický vývoj, tak súčasne vnímanú formu. Nasledovalo predstavenie disciplíny ontologického inžinierstva, ktoré by malo pomáhať úspešne vyvíjať ontológie. Aj keď sa táto disciplína snaží „kopírovať" už pomerne vyspelé softwarové inžinierstvo (metodológie, postupy, návrhové vzory, atď.), to ontologické ešte stále nedosahuje jeho úroveň. Aj napriek snahe zakomponovať do práce priblíženie oboru reprezentácie znalostí, jeho históriu a súčasnosť, ktorý viedol k vytvoreniu niečoho ako je ontológia, bola táto časť prace kvôli zachovaniu rozumne dlhého textu z práce nakoniec vyškrtnutá. Teoretická časť bola zavŕšená priblížením súčasnej modernej a rýchlo rozvíjajúcej sa technológií sémantického webu, ktorý ako jeden zo základných stavebných kameňov využíva práve ontológie.

V praktickej časti sme si predstavili jednu v veľkých reálne používaných Ontológií SNOMED CT. Natívny formát uloženia tejto ontológie je v proprietárnej štruktúre zloženej z prepojených textových súborov s *holým textom*[212]. OWL forma je generovaná pomocou Phyton skriptu. Táto skutočnosť žiaľ SNOMED CT od-nominovala z pozície ontológie vhodnej pre predstavenie typickej ontológií.

V ďalšej časti práce sme sa venovali návrhu aplikácie pre správu a vizualizáciu ontológií. V prieskume predchádzajúcom tento proces sme si vyskúšali niekoľko súčasných (asi top) nástrojov, ktoré už podobnému účelu slúžia a sú už pomerne vyspelé. Aby aplikácia priniesla niečo nové, rozhodli sme sa typ webovej aplikácie. Nasledoval prieskum existencie komponent a nástrojov (prevažne open-source), ktoré sú nutné pre návrh a implementáciu takejto aplikácie. Sklamaním pre mňa ako .Net vývojára bolo zistenie, že táto sféra zatiaľ praje iba Java vývojárom, pre ktorých asi jediných existuje dostatočné množstvo vyskúšaných a dobre zdokumentovaných frameworkov, úložísk a aplikačných programových rozhraní (API) na prácu s ontológiami. Pri návrhu sa znova ukázalo, že dôkladná analýza ušetrí veľké množstvo času implementácie. Malá prax a znalosť celkovej problematiky ma viedla na výber OWL API ako komponenty pre prácu s ontológiami. Neskoršie uvedomenie si neexistencie napojení na perzistetné úložiská a nemožnosť práce so SPARQL viedlo k výmene tejto komponenty za komplexnejší framework Jena. Z rozhodnutia webovej viac-užívateľskej formy aplikácie vyplynula celá rada funkcionality, ktorú so sebou tento typ aplikácie nesie a tým zaťažuje samotný vývoj a návrh aplikácie. Tento fakt, ako aj neznalosť

[212] Holý text anglicky *Plain text*

prostredia Javy a v podstate učenie sa problematiky sémantických webov a ontológií písaním samotnej práce, nenechal veľa priestoru pre samotnú implementáciu prototypu navrhovanej aplikácie. K tej existuje v prílohách software design dokument (viz. 13 Dodatok A: CD13) a prvá verzia aplikácie s prvými funkcionalitami (viz. 13 Dodatok A: CD).

9. Bibliografia

1. **McCarthy, J.** Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence.* 1980, s. 13:27–39.

2. **Gruber, T. R.** A translation approach to portable ontology specifications. *Knowledge Acquisition.* 1993, s. 5(2):199–220.

3. —. Toward principles for the design of ontologies used for knowledge sharing. [ed.] N. In Guarino a R., Poli. *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Deventer : Kluwer Academic Publishers, 1994.

4. *Ontologies: principles, methods, and applications.* **Uschold, M. a Grüninger, M.** 1996, Knowledge Engineering Review, s. 11(2):93–155.

5. **Gruber, T. R.** Formal Ontology in Conceptual Analysis and Knowledge Representation. [ed.] N. In Guarino a R. Poli. *Toward principles for the design of ontologies used for knowledge sharing.* Deventer : Kluwer Academic Publishers, 1994.

6. **Lakoff, G.** *Women, Fire and Dangerous Things.* s.l. : University of Chicago Press, 1987.

7. **Ogden, C. K.** *Basic English: A General Introduction with Rules and Grammar.* 1930.

8. **Breuker, J. a Van De Velde, W.** *CommonKADS Library for Expertise Modeling: reusable problem solving components.* Amsterdam/Tokyo : IOS-Press/Ohmsha, 1994.

9. *Using explicit ontologieslo for kbs development.* **van Heijst, G., Schreiber, A. T. a Wielinga, B.** 1997, International Journal of Human-Computer Studies, s. 46(2/3):183–292.

10. *DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO).* **Oberle, D., Ankolekar, A. a al., et.** 2007, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, s. 5:156–174.

11. **Ghilardi, S., Lutz, C. a Wolter, F.** *Did i damage my ontology? a case for conservative extensions in description logics.* [ed.] P. In Doherty, J. Mylopoulos a C. A. Welty. s.l. : AAAI Press, 2006. s. 187–197.

12. **Hoekstra, R.J.** *Ontology Representation : design patterns and ontologies that make sense.* Amsterdam : s.n., 2009.

13. *Knowledge patterns.* **Clark, P., Thompson, J. a and Porter, B.** [ed.] A. In Cohn, F. Giunchiglia a B. Selman. s.l. : CA. Kaufmann, 2000. Proceedings of the 7th International Conference KR' 2000. s. 591–600.

14. **IHTSDO, et al.** *SNOMED CT® User Guide July 2011 International Release (US English).* s.l. : ©2002-2011 The International Health Terminology Standards Development Organisation (IHTSDO), 2011.

15. **McCarthy, J a Hayes, J. P.** *Some philosophical problems from the standpoint of artificial intelligence.* 1986.

16. **Quillian, R. M.** *Semantic memory.* Cambridge, Massachusetts : Bolt Beranek & Newman, 1966.

10. Zoznam tabuliek

11. Zoznam Obrázkov

12. Zoznam použitých skratiek

Skratka	Význam	Anglický originál
PC	Počítač	Personal Computer
Stránka	Internetová stránka	Internet Site
Web	Internet	World Wide Web
UI	Umelá inteligencia	Artificial Intelligence
RZ	Reprezentácie Znalostí	Knowledge Representation
ZI	Znalostné inžinierstvo	Knowledge Engineering
OWL		Web Ontology Language
ZZ	Získavanie znalostí	Knowledge Acquisition
SSI	Systém spracovania informácií	Information Processing System
EIP	Elementárny informačný proces	Elementary Information Process
PP	Produkčné pravidlá	Production Rules
KRL	Jazyk na reprezentáciu znalostí	Knowledge Representation Language
SI-Nets		Structured Inheritance Networks
SZB	Systém založený na znalostnej báze	Knowledge Based System
DL	Deskriptívna logika	Descriptive Logic
URI		Universal Resource Identifier
URL		Uniform Resource Locator
URN		Uniform Resource Name
XML		eXtensible Markup Language
RDF		Resource Description Framework
RDFS		Resource Description Framework Schema

Skratka	Význam	Anglický originál
SHOE		Simple HTML Ontology Extension Language
FW	Všeobecná zjednocujúca základná kostra	Framework
ODP	Ontologický návrhový vzor	Ontology Design Pattern
DT	distribučná tabuľka pre jeden z typov komponent, konceptov, popisov, vzťahov koncept – Concept	
SCTID	SNOMED CT ID	SNOMED CT ID
CHT	Historická tabuľka komponent	Component History Table
RT		Relationship Table
SCT	SNOMED CT	SNOMED CT
SRT		Stated relationship table
OMS	Systém pre správu ontológií	Ontology Management System
FSN	Plne špecifikované meno	Fully Specified Name
PT	Preferovaný výraz	Preferred Term
SWS	Zásobník sémantického riešenia	Semantic web stack
Browser	Internetový prehliadač	Internet browser
OMS	Manažment systém pre ontológie	Ontology management system
ACL	Zoznam oprávnení k objektom	Access Control List
CRT	Referenčná tabuľka komponent	Component references table
RID	Identifikátor referencie	ReferenceId
ST	Tabuľka podmnožín	Subset table
SID	Identifikátor podmnožiny	SubsetID

Skratka	Význam	Anglický originál
OID	Pôvodný identifikátor podmnožiny	OriginalId
SMT	Členská tabuľka podmnožín	Subset member table
CMST	Tabuľka krížových máp	Cross Map Sets Table
CMT	Tabuľka krížovej mapy	Cross Maps Table
CMTT	Tabuľka cieľov krížovej mapy	Cross Map Targets Table
SCTID		SNOMED CT Indetifier
SRT	„stated" vzťahová tabuľka	Stated relationship table
OGP		Open Graph Protocol
OMS	Manažment systému pre ontológie	Ontology management system
ACL		Access control list
Browser	Internetovom prehliadači	
GVF	grafovo vizualizačný framework	
SDD	Software design dokument	Software design documnet

13. Dodatok A: CD

Priložené CD obsahuje:

- dokumenty\Software_design_dokument.pdf – software design dokument navrhovanej aplikácie
- dokumenty\Diplomova_prava.pdf – text diplomovej práce v pdf formáte
- aplikacia\Tutorial.pdf – návod na spojazdnenie prototypu aplikácie a krátky užívateľský návod
- aplikacia\play\ – adresár obsahujúci súbory play! frameworku
- aplikacie\play_projects\test_06\ - kompletné súbory projektu (vrátane projektových súborov IntelliJ IDEA 13.1.1 [213]vývojového prostredia) prvej verzie prototypu aplikácie

[213] Domáca stránka IntelliJ IDEA studia: http://www.jetbrains.com/idea/